P. 1877/80

12 1980

informatyka

WYDAWNICTWO NOT SIGMA

Poszukujemy współpracowników

W ciągu ostatnich miesięcy sytuacja w Polsce zmieniła się tak dalece, że realne stają się nawet takie sfery działań, których wcześniej nie sposób było nawet przewidywać. Praca naszej Redakcji również może zmienić swój charakter; INFORMATYKA ma teraz szansę stać się autentycznym forum środowiska, służącym wymianie istotnych poglądów i doświadczeń.

Rządy autorytatywne w Polsce, które spowodowały w efekcie m. in. całkowite uzależnienie informatyków od decyzji odgórnych (częściej politycznych niż gospodarczych), ograniczały do niedawna zakres aktywności pisma. Redakcja nie była choćby dopuszczana do źródeł prawdziwych informacji o informatyce, mogła się opierać jedynie na spreparowanych zestawieniach, przygotowanych przez poszczególne instytucje dla pokrzepienia decydentów. Ranga pisma — w następstwie — znacznie malała.

Brak ścisłego związku informatyki ze społecznym i gospodarczym stanem kraju przyniósł w jej ramach dość swobodny rozwój prac naukowych, dający w znacznej mierze świadectwo niefrasobliwości „twórców". Niedostatek jednoznacznych kryteriów oceny tych prac zwiększył obfitość dokonań pseudonaukowych. Tym piętnem naznaczona została również INFORMATYKA.

Większość przedstawicieli środowiska, którzy byli świadomi realiów, nie podjęła współpracy z pismem, sądząc zapewne, że ich głos i tak zostanie odrzucony przez Redakcję, reprezentującą interesy instytucji zarządzających, albo obawiając się przykrych reperkusji. Ich postawa utorowała drogę na łamy wielu pracom o niskiej wartości, które nie znalazły silnej konkurencji.

Dzisiaj, kiedy prasa zaczyna zbliżać się coraz bardziej do rzeczywistości, kiedy skutki wieloletniej irracjonalnej gospodarki, nauki i techniki zostają poddane ostremu społecznemu osądowi — pojawia się szansa odrzucenia w przeszłość złych doświadczeń. Ale — powtarzamy jeszcze raz — zależy to przede wszystkim od ś r o d o w i s k a. Jeśli ci informatycy, których poglądy i wiedza godne są upowszechnienia, nie zdecydują się na współpracę ze s w o i m pismem, nigdy nie zaspokoi ono ich oczekiwań. A oczekiwania te — jak wiemy — są duże.

Wzywamy zatem wszystkich informatyków, którzy czują się na siłach przedstawiać na łamach swoje poglądy i przekazywać swą wiedzę, by niezwłocznie podjęli współpracę z INFORMATYKĄ. Czekamy na kontakt. Zasady współdziałania ustalimy z każdą osobą indywidualnie. Oferujemy wszystkie formy stałej współpracy. Publikacje będą oczywiście honorowane zgodnie z obowiązującymi stawkami autorskimi. Prosimy o szybką decyzję.

REDAKCJA

WYDAWNICTWO SIGMA
NACZELNA ORGANIZACJA TECHNICZNA
CZASOPISM I KSIĄŻEK TECHNICZNYCH
ul. Świętokrzyska 14a
00-950 Warszawa
skrytka pocztowa 1004

KOLEGIUM REDAKCYJNE

Redaktor naczelny: prof. dr hab. Leon ŁUKASZEWICZ

dr Krystyn BERNATOWICZ, prof. dr hab. inż. Konrad FIAŁKOWSKI (zastępca redaktora naczelnego), doc. Zbigniew GACKOWSKI, mgr inż. Zbigniew GLUZA, dr Janusz GWIAZDA, mgr inż. Stanisław JASKÓLSKI, Władysław KLEPACZ (zastępca redaktora naczelnego), mgr inż. Wincenty ŁADA, dr inż. Tomasz PAWLAK, mgr inż. Antoni WIESNOWSKI

Sekretarz redakcji: mgr Teresa JABŁOŃSKA Red. techn.: Ewa KAMIŃSKA

RADA PROGRAMOWA

Prof. dr hab. Tadeusz PECHE (przewodniczący), mgr inż. Tomasz BAŃKOWSKI (sekretarz), mgr inż. Antoni BOSSOWSKI, mgr inż. Roman BURNO, prof. dr hab. Andrzej JANICKI, mgr inż. Jan KRAMARCZUK, prof. dr hab. inż. Juliusz KULIKOWSKI, prof. dr hab. Leon ŁUKASZEWICZ, gen. dr inż. Marian PASTERNAK, mgr inż. Bronisław PIWOWAR, mgr Zbigniew SUBSTYK, mgr Jerzy TRYBULSKI, doc. dr hab. Tadeusz WALCZAK, dr inż. Jan ŻYDOWO

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pokój 326, tel. 27-71-40, dyżury redakcji 10.00—13.00

Zakł. Graf. „Tamka". Zam. 468. Papier druk. sat. V kl. 70 g. A1. Obj. 5 ark. druk. Nakład 7000 egz. O-36.

Cena egzemplarza zł 30.— IDNEKS 36124 Prenumerata roczna zł 360.—

Gackowski Z.: Perspektywy komputeryzacji badania i projektowania systemów. Część 1: język

INFORMATYKA 1980, nr 12, s. 4

Przegląd obecnego stanu rozwoju technik wspomaganego komputerem projektowania systemów zarządzania. W części pierwszej autor omawia sposoby formalizacji opisu takich systemów za pomocą odpowiedniego języka, a także wyniki porównania możliwości różnych tego rodzaju języków.

Гацковски З.: Перспективы автоматизирования исследования и проектирования систем. Часть I: язык

ИНФОРМАТИКА 1980, № 12, стр. 4

Просмотр настоящего состояния развития техники проектирования систем вспомогаемой вычислительной машиной. В первой части автор обсуждает способы формализации описания таких систем при помощи соответствующего языка и результаты сравнения возможностей разных языков этого рода.

Rzeczkowski W., Subieta K.: LINDA — system zarządzania bazą danych. Część 2

INFORMATYKA 1980, nr 12, s. 8

Część druga artykułu na temat systemu zarządzania bazą danych LINDA zawierająca poparte przykładami szczegóły charakterystyki budowy systemu oraz możliwości zastosowania użytego języka manipulacji danymi. W zakończeniu podano ocenę wartości użytkowej systemu na podstawie dotychczasowych doświadczeń eskploatacji.

Жечковски В., Субета К.: ЛИНДА — система управления базой данных. Часть II

ИНФОРМАТИКА 1980, № 12, стр. 8

Часть вторая статьи касающаяся системы управления базой данных ЛИНДА содержит, потвержденные примерами, подробности характеристики построения системы и возможности использования применяемого языка манипуляции данными. В конце статьи дается оценка эксплуатационной ценности системы на основе до сих пор проведенных эксплуатационных опытов.

Kornacki W.: Wybrane modele niezawodnościowe programów

INFORMATYKA 1980, nr 12, s. 11

Charakterystyka ważniejszych modeli stosowanych do oceny niezawodności programów oraz propozycji modelu rozszerzonego, pozwalającego na głębszą i bardziej preyzyjną ocenę niezawodności.

Корнацки В.: Избранные модели надежности программ

ИНФОРМАТИКА 1980, № 12, стр. 11

Характеристика важнейших моделей применяемых для оценки надежности программ и расширенного моделя, дающего возможность более глубокой и более точной оценки надежности.

Chrobot S.: System operacyjny SOM-51 dla minikomputera MERA 400

INFORMATYKA 1980, nr 12, s. 15

Szczegółowa charakterystyka pierwszej handlowej wersji systemu operacyjnego SOM-5, opracowanego w Przedsiębiorstwie Systemów Komputerowych MERA-SYSTEM w Warszawie.

Хробот С.: Оперативная система СОМ-51 для малой вычислительной машины МЕРА 400

ИНФОРМАТИКА 1980, № 12, стр. 15

Подробная характеристика первой версии оперативной системы СОМ-5 разработанной в Предприятии автоматизированных систем МЕРА-СИСТЕМ в г. Варшава.

Pawelec B.: Komputerowy system automatyzacji eksperymentów laboratoryjnych

INFORMATYKA 1980, nr 12, s. 23

Charakterystyka systemu automatyzacji eksperymentów laboratoryjnych opracowanego i zbudowanego w Zakładach MERA-ELWRO w oparciu o komputer ODRA 1325 oraz sprzęt ODRA-SMA (System Modułów Automatyki).

Павэлец Б.: Вычислительная система автоматизации лабораторийных экспериментов

ИНФОРМАТИКА 1980, № 12, стр. 23

Характеристика системы автоматизации лабораторийных экспериментов, разработанной и созданной в Предприятии МЕРА-ЭЛЬВРО на основе вычислительной машины ОДРА 1325 и оборудования ОДРА-СМА (Система Модулей Автоматики).

Gackowski Z.: Prospects of systems research and design computerization. Part 1: the language

INFORMATYKA 1980, No 12, p. 4

Actual development overview of the computer assisted management system design. In the first part of the article the author presents methods of such system description using suitable language, as well comparison results of different languages possibilities.

Gackowski Z.: Die Automatisierungsaussichten der Systemforschung- und Projektierung. Teil 1: die Sprache

INFORMATYKA 1980, Nr. 12, S. 4

Die Übersicht des heutigen Entwiclungszustandes von rechnergestützter Projektierung der Verwaltungssysteme. Im ersten Teil bespricht der Autor die Formalisierungsmethoden von Beschreibung solcher Systeme mittels einer entsprechenden Sprache, sowie die Vergleichungsergebnisse von Möglichkeiten verschiedener Sprachen solcher Art.

Rzeczkowski W., Subieta K.: LINDA — a data base management system. Part 2

INFORMATYKA 1980, No 12, p. 8

Second part of the article on data base management system LINDA containing with examples supported details of the implemented data manipulation language. In the conclusion presented evaluation of the system, based on the hitherto operation experience.

Rzeczkowski W., Subieta K.: LINDA — ein Datenbasisverwaltunssystem. Teil 2

INFORMATYKA 1980, Nr. 12, S. 8

Zweiter Teil des Artikels über das Datenbasisverwaltungssystem LINDA, der die mit Beispielen gestützte Einzelheiten der Systemstruktur und die Anwendungsmöglichkeiten der verwendeten Datenmanipulationssprache enthält. Im Schlussteil wurde die Gebrauchswertschätzung des Systems auf Grund der bisherigen Betriebserfahrung angegeben.

Kornacki W.: Selected program reliability models

INFORMATYKA 1980, No 12, p. 11

Characteristics of more significalnt models applied for programs reliability evaluation, as well proposal of the extended model, which enables the more intense and precise reliability evaluation.

Kornacki W.: Ausgewählte Modelle für die Programmezuverlässigkeit

INFORMATYKA 1980, Nr. 12, S. 11

Die Charakteristik der wichtigsten Modellen, die für die Programmezuverlässigkeit angewendet werden, sowie die Charakteristik eines vorgeschlagenen erweiterten Modells, das die vertiefte und mehr präzise Zuverlässigkeitbeurteilung ermöglicht.

Chrobot S.: Operating system SOM-51 for the MERA 400 minicomputer

INFORMATYKA 1980, No 12, p. 15

Detailed characteristics of the first marketable version of the SOM-5 operating system, elaborated by the Computer Systems Enterprise MERA-SYSTEM in Warsaw.

Chrobot S.: Das Betriebssystem SOM-51 für den MERA 400 Kleinrechner

INFORMATYKA 1980, Nr. 12, S. 15

Genaue Charakteristik der ersten Marktversion des Betriebssystems SOM-5, die im Unternehmen für Computersysteme MERA-SYSTEM in Warschau erarbeitet wurde.

Pawelec B.: Computer system for laboratory experiments automation

INFORMATYKA 1980, No 12, p. 23

Characteristics of the computer system for laboratory experiments automation, designed and built in MERA-ELWRO works using ODRA 1325 computer and ODRA-SMA hardware (Automation Modules System).

Pawelec B.: Das Rechnersystem für die Automatisierung von Laborexperimenten

INFORMATYKA 1980, Nr. 12, S. 23

Die Charakteristik des Automatisierungsystems für Laborexperimente, das in MERA-ELWRO Werke mit Ausnutzung des ODRA 1325 Rechners und des ODRA-SMA Hardwers (System für Automatikmodule) erarbeitet und gebaut wurde.

Informatyka

zastosowania w gospodarce, technice i nauce

Nr 12
MIESIĘCZNIK
1980
ROK XV
Grudzień

ORGAN KOMITETU INFORMATYKI, MINISTERSTWA NAUKI, SZKOLNICTWA WYŻSZEGO I TECHNIKI ORAZ KOMITETU NAUKOWO-TECHNICZNEGO NOT DS. INFORMATYKI

P.1877/80

W NUMERZE: Strona

Perspektywy komputeryzacji badania i projektowania systemów (język) 4
Zbigniew Gackowski

LINDA — system zarządzania bazą danych. Część 2 8
Wiktor Rzeczkowski, Kazimierz Subieta

Wybrane modele niezawodnościowe programów 11
Witold Kornacki

System operacyjny SOM-51 dla minikomputera MERA 400 15
Stanisław Chrobot

Komputerowy system automatyzacji eksperymentów laboratoryjnych 23
Bronisław Pawelec

SPIS TREŚCI ROCZNIKA 1980 19

Z KRAJU

ZJEDNOCZENIE INFORMATYKI

Egzekutory z usprawnieniami wyprowadzania wydruków 25
Leonard Dobiecki, Zofia Guryn

Po raz ostatni w Jeleniej Górze? *(KB)* 27

Złaz Informatyków *(IK)* 27

INTERBIURO'80 *(ZG)* 27

Informatyka w badaniach naukowych chemii 28
Janusz Zalewski

ZWIĄZKI ZAWODOWE

Niezależni i samorządni 31
Włodzimierz Grudziński, Ryszard Kasprzyk

Związek czy Stowarzyszenie? 32

LISTY 32

ZE ŚWIATA

Szanse i granice przetwarzania informacji 33
Józef Piasecki

RECENZJE

Kontrola w nowoczesnych systemach finansowo-księgowych 36
Maria Waśko

TERMINOLOGIA

O jednolitą terminologię: „Programowanie" 36
Janusz Zalewski

POGLĄDY

Gdzie jest informatyka? 39
Oprac. Z. Gluza

Cz.3573

ZBIGNIEW GACKOWSKI
Instytut Organizacji Zarządzania i Doskonalenia Kadr
Warszawa

Perspektywy komputeryzacji badania i projektowania systemów (język)

Zapotrzebowanie na skomputeryzowane systemy informacyjne zarządzania narasta w świecie lawinowo. Wzrastające możliwości komputerów umożliwiają tworzenie nowych systemów oraz modyfikację istniejących. Odmiennie od innych systemów działania, systemy informacyjne zarządzania nie mogą być produkowane masowo, lecz muszą być projektowane na miarę potrzeb jednostek organizacyjnych, którym mają służyć. Sytuacja taka wymaga dysponowania sprawnymi technikami [1]) projektowania. Obserwuje się na tym odcinku wiele wysiłków mających na celu udoskonalenie zarówno samego procesu jak i produktu projektowania. Jeden z bardziej perspektywicznych kierunków rozwoju opiera się na obserwacji, że projektowanie systemów informacyjnych zarządzania, jak zresztą i wszelkich innych zamierzonych systemów działania, jest procesem przetwarzania informacji. Jak dotąd, procesu tego nie można całkowicie zautomatyzować, a tylko wspomagać komputerem jego przebieg.

Ostatnio obserwuje się szybkie postępy automatyzacji prac inżynierskich, lecz szczególnie duże perspektywy rysują się w dziedzinie komputerowego wspomagania projektowania systemów informacyjnych, mimo że nie należy ona do metodologicznie najbardziej zaawansowanych. Jest to wynikiem pewnego korzystnego zbiegu okoliczności. Mianowicie, systemy informacyjne projektowane są przez specjalistów, którzy najlepiej rozumieją możliwości współczesnych komputerów i fachowo najlepiej są przygotowani do ich stosowania. Osiągnięte wyniki pozwalają już wyraźnie dostrzec realne możliwości szybkiego rozwinięcia wspomaganych komputerem technik badania i projektowania tych systemów działania, dla których można stworzyć adekwatny operacyjny model ich funkcjonowania.

W niniejszym artykule zostaną przedstawione prowadzone w tym zakresie najbardziej perspektywiczne prace rozwojowe. Opis ich oparty jest na danych uzyskanych z ISDOS Project [2]) w wyniku dotychczasowej współpracy autora artykułu z tym zespołem.

WSTĘPNE WARUNKI

W oparciu o przegląd różnych technik przygotowania systemów informacyjnych [3, 19, 22, 23] określił, że wspomagana komputerem technika projektowania systemów informacyjnych powinna umożliwiać:

- wyrażenie informacji o dużych projektowanych obiektach w formie przetwarzalnej przez komputer
- utrzymywanie ich w komputerowej bazie danych
- otrzymywanie ich na wyjściu komputera w różnych formach i układach
- nadzorowanie przebiegu projektowania.

Można przyjąć, że kompletna technika wspomaganego komputerem projektowania składa się z czterech zasadniczych części:

- **modelu koncepcyjnego,** stanowiącego podstawę dla tworzenia szczególnego przypadku projektowanego obiektu zgodnie z postawionymi wymaganiami
- **metody specyfikującej** czynności projektowania, porządek ich wykonywania, konieczne funkcje koordynacyjno-kontrolne oraz dokumenty wynikowe
- **języka albo notacji** stosowanej dla przedstawienia projektowanego obiektu
- **pomocy i narzędzi** umożliwiających i ułatwiających projektowanie według przyjętej metody.

Podstawą dla jakiejkolwiek techniki projektowania jest koncepcyjny model istoty projektowanego obiektu. Jeżeli informacja o tym modelu ma być przetwarzana przez komputer, to wszystkie jego elementy powinny być w pełni operacyjnie określone.

Odrębnym i wydaje się najtrudniejszym zagadnieniem jest algorytmizacja procesu projektowania. Samemu zagadnieniu kolejności rozstrzygania w procesie projektowania o kształcie i własnościach projektowanego obiektu w zakresie wszystkich części i aspektów jego charakterystyki, dla wszelkich systemów działania jak i dla systemów informacyjnych, poświęcono wielką liczbę prac. Nadal jednak daleko jest do jednomyślności poglądów. Doświadczenia i intuicja projektanta zapewne jeszcze przez długi czas pozostaną czynnikami niezastąpionymi, chociaż dziś mogą one być już wspomagane przez komputer, na przykład dzięki możliwości sprawniejszego analizowania różnych wariantów rozwiązań. Stąd, coraz większego znaczenia nabierają wszelkie metody wspomaganego komputerem modelowania rozwiązań, także w przypadku systemów informacyjnych [3]).

Tradycyjne techniki posługują się rysunkami, diagramami, tablicami, listami, formułami matematycznymi oraz tekstem w naturalnych językach, zaś formalnie zapisany opis uzupełniany jest jeszcze informacjami, które pozostały w umyśle projektanta. Cechuje je wiele niedogodności: trudność sprawdzania kompletności i zgodności wewnętrznej opisu, wysokie koszty modyfikowania w miarę zachodzących zmian z uwagi na konieczność wielokrotnego przepisywania znacznej części opisu, wreszcie nieunikniona wieloznaczność [22].

W celu udoskonalenia opisu systemów informacyjnych opracowuje się i proponuje coraz to nowe metody prezentacji. Rozwijają się one w kierunku zastępowania naturalnych języków bardziej formalnymi, zaś jednoznaczność graficzną prezentacji poprawia się przez wprowadzenie udoskonalonych reguł sporządzania schematów i rysunków. Jones 12 podzielił wszelkie języki na sznurowe — operu-

[1]) — Termin technika używany jest tu w znaczeniu celowego posługiwania się określoną metodą i wymaganymi przez nią środkami dla wykonania zamierzonego zadania

[2]) — ISDOS (Information System Design and Optimalization System) Project — przedsięwzięcie badawczo-rozwojowe realizowane w The University of Michigan (USA)

[3]) Między innymi: Alford [1], Board, Pigneur [2], Spewak [13], Willis [24]

jące ciągami znaków ang. *(string languages)* oraz graficzne. Badania porównawcze wielu spośród proponowanych języków i metod opisów [22] [4]) wykazały, że żadne z nich nie mają wyraźnej przewagi. W ostatecznym rachunku każdy opis służy do komunikowania komuś czegoś, a więc czytelność i zrozumiałość jest wymaganiem o najwyższym priorytecie, przed którym wszystkie inne muszą ustąpić. Wyróżniono tam aż osiem typów odbiorców opisów, w tym i same komputery. Z kolei, wymagania każdego indywidualnego użytkownika mogą dodatkowo różnić się pod względem ich formy i poziomu. Tak więc, opis systemu informacyjnego musi być kombinacją różnych prezentacji. Na szczęście, współczesna technika komputerowa nie utrudnia, lecz ułatwia równoległe stosowanie różnych form opisu, odmiennych na wejściu, innych w pamięci komputera przechowującego opis, a jeszcze innych na wyjściu komputera. Człowiek został przy tym całkowicie uwolniony od wykonywania wszelkich przekształceń przy przechodzeniu od jednej formy do innej.

Oczywiście wymaga to z kolei opracowania całego systemu programów umożliwiających wykorzystanie komputerów do jak najdalej idącego wyręczania ludzi pracujących nad rozwojem nowych systemów informacyjnych.

MODELE I JĘZYKI OPISU SYSTEMÓW

Opierając się na porównaniach bardzo wielu proponowanych na świecie modeli i języków opisu systemów informacyjnych [20] w ISDOS Project, od samego zarania prowadzonych tam prac rozwojowych, intuicyjnie skłaniano się do wzorowania się przede wszystkim na języku naturalnym, w tym przypadku angielskim. Niewątpliwie, było to również podyktowane naciskiem organizacji użytkujących i finansujących projekt (sponsorów) na uzyskanie wymaganej komunikatywności wyników. Intuicja ta była jednak podbudowana również naukowym uzasadnieniem. Systemy informacyjne są na ogół obiektami o szczególnie dużej złożoności, trudnej do całościowego objęcia w ludzkim umyśle. Opis systemu informacyjnego, rzeczywistego czy projektowanego, powinien być możliwie wiernym odbiciem takiego obiektu i to odbiciem możliwie najlepiej nadającym się do zakomunikowania ludziom współpracującym nad projektem.

Człowiek zawsze odczuwał potrzebę zrozumienia otaczającej go nader złożonej rzeczywistości. W trakcie naturalnej ewolucji umysł ludzki nie stworzył w tym celu nic bardziej doskonałego aniżeli naturalny język, w którym opisuje otaczające go zjawiska a następnie komunikuje to innym. A więc, w budowie języków naturalnych, niezależnie jak bardzo niedoskonałe one nam się wydają, muszą być zawarte wszystkie, aktualne — na obecnym etapie rozwoju człowieka — pojęcia pierwotne, za pomocą których można opisać najbardziej złożony obiekt i treść tego opisu zakomunikować w sposób zrozumiały innym ludziom.

Budowa każdego naturalnego języka jest na wskroś relacyjna. Każde zdanie, które jest najmniejszą konstrukcją językową zdolną bez kontekstu do samodzielnego zakomunikowania czegoś, zawsze orzeka o zachodzeniu relacji między:

- obiektami
- relacjami i ich atrybutami
- relacjami między obiektami i atrybutami tych relacji.

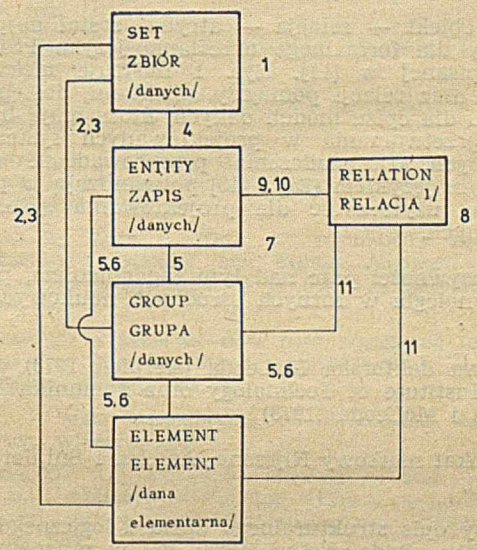
Oczywiście, w teoriomnogościowym ujęciu [6] relacje między obiektami a ich własnościami (atrybutami) przedstawić można jako jednoargumentową relację w zbiorze Z obiektów ($e \in R$ przy $R \subseteq Z$) oraz analogicznie relację między relacjami — jako jednoargumentową relację w zbiorze relacji między obiektami. Praktyczna własność takiego ujęcia polega na tym, że wszystkie wymienione przypadki relacji można przedstawić jako:

- zbiory pojedynczych elementów dla atrybutów obiektów
- zbiory par elementów dla relacji między obiektami oraz atrybutów relacji
- zbiory 2,3... n elementów dla relacji wieloczłonowych.

Nie oznacza to jednak, że ludzie w ten sposób opisują obiekty i wyniki swoich obserwacji komunikują innym. Języki naturalne wykształciły swoją budowę przede wszystkim dla potrzeb komunikowania się.

Tak zrodziła się pierwotna idea budowy języka PSL (Problem Statement Language) [20], a następnie hipoteza naukowa, że wiele modeli będących (implicite) podstawą istniejących technik projektowania systemów informacyjnych może być uważanych za specjalne przypadki obiektów, relacji i atrybutów. Mogą one więc być reprezentowane przez uogólniony model obiekt — relacja — atrybut (Entity Relation ship Attribute (ERA) Model) oparty na pracach Chena [4] oraz Davenporta [5]. Model ten [23] teoretycznie wystarcza. Jeżeli jednak ma on posłużyć do budowy łatwych w użyciu języków i technik projektowania wspomaganego komputerem, zgodnych także z innymi wstępnie sformułowanymi wymaganiami ze względu na zdolność komunikowania się, powinien być uzupełniony o następujące elementy pomocnicze:

- objaśniające komentarze wolnotekstowe, które można uważać za specjalne atrybuty składające się z dowolnej liczby znaków



Rys. 1. Model obiekt — relacja — atrybut dla części języka PSL (Problem Statement Language) wg Teichroew i in. (październik, 1979)

Legenda:

Wszystkie relacje w języku PSL reprezentowane przez numerowane linie łączące prostokąty typów obiektór są typu „wiele do wielu"

1. SUBSETS (ZBIÓR może być podzbiorem innego ZBIORU danych)

2. ORDERS (GRUPA danych lub ELEMENT (dana elementarna) może porządkować ZBIÓR)

3. SUBSETTING CRITERIA FOR (GRUPA lub ELEMENT może być kryterium podziału ZBIORU na podzbiory)

4. COLLECTION OF (ZBIÓR może być kolekcją ZAPISÓW)

5. CONSIST OF (ZAPIS lub GRUPA mogą składać się z GRUP lub ELEMENTÓW)

6. IDENTIFIES (GRUPA lub ELEMENT mogą identyfikować GRUPĘ lub ZAPIS)

7. SUBPARTS ARE (ZAPIS może posiadać ZAPISY jako podzapisy)

8. COMPOSED OF (RELACJA między ZAPISAMI może być złożona z innych RELACJI *)

9. LEFT PART IS (ZAPIS może byś lewą stroną RELACJI *)

10. RIGHT PART IS (ZAPIS może być prawą stroną RELACJI *)

11. ASSOCIATED DATA IS (ELEMENTY lub GRUPY mogą być związane z RELACJAMI *)

*) **UWAGA:** RELACJA (w prostokącie) jest relacją, która może zachodzić między ZAPISAMI i jest traktowana jako typ obiektu w języku PSL, który może być definiowany samodzielnie przez użytkownika lub projektanta systemu informacyjnego — w zależności od potrzeb. Różni się ono od relacji standardowych przewidzianych z góry w języku PSL, co do których użytkownik czy projektant mogą jedynie orzec, że zachodzą między określonymi obiektami, nie mogą ich jednak indywidualnie określać, np. relacje 1—11 omówione wyżej oraz inne — nie omówione.

[4]) — Patrz także: Z. Gackowski — Problemy komputeryzacji dokumentowania systemów informacyjnych. Organizacja, Metody, Technika z. 1 (241) styczeń 1979, s. 31 — Użytkownicy i przeznaczenie dokumentacji

- wyrazy kluczowe o specjalnym znaczeniu w danym języku ze względu na jego komputerowe przetwarzanie (czyli komunikację z komputerem), które nie mogą być używane do innych celów

- wyrazy uzupełniające, które mogą być używane wszędzie gdzie poprawi to czytelność (komunikatywność) języka

- zdania [5], będące kombinacją nazw obiektów (nadanych im przez użytkownika), słów kluczowych, atrybutów i wartości atrybutów, które służą dla określenia konkretnego przypadku zachodzenia relacji.

Tak rodziły się początki uogólnionego modelu relacyjnego metajęzyka. W oparciu o taki model można w, zależności od specyficznych potrzeb półautomatycznie konstruować języki specjalizowane do opisu różnych aspektów różnych systemów działania.

Oczywiście model metajęzyka, modele konkretnych języków zdefiniowanych w oparciu o ten model, a także sam opis systemu sporządzony przy użyciu konkretnie zdefiniowanego języka mogą być odwzorowane w formie odpowiednich grafów, te zaś z kolei — w formie zbiorów 2,3... n elementów lub macierzy incydentnych [5].

Model obiekt — relacja — atrybut został najpierw zastosowany dla formalnego określania języka PSL w jego wersji opisanej w [11]. Rys. 1 ilustruje niektóre typy obiektów oraz relacji pomiędzy obiektami, dostępne w języku PSL dla opisu modeli danych, które mają być przedmiotem przetwarzania w projektowanych systemach informacyjnych. Doświadczenia przeprowadzone na dużą skalę z tym językiem [25] i [13] potwierdzają w pełni jego praktyczną użyteczność dla projektowania systemów informacyjnych.

O intensywności prac nad tym zagadnieniem, prowadzonych równolegle w różnych ośrodkach naukowych, świadczą:

- rozprawa doktorska McLeoda (sierpień 1978) w Massachusetts Institute of Technology oraz komunikat naukowy Hammera i McLeoda (1978)

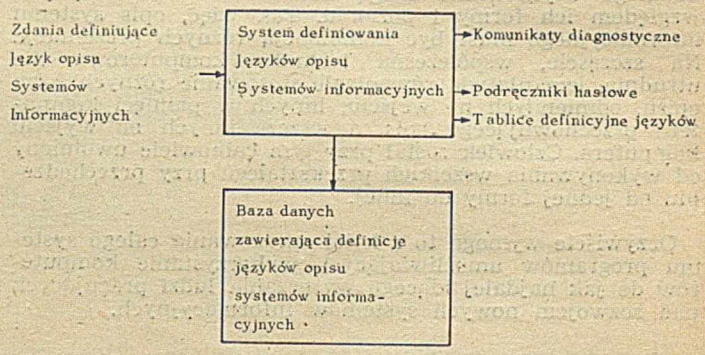
- komunikat naukowy Nijssena, Assch'a i Snijdersa (kwiecień 1979),

- wstępny opis strukturalnego opisu i logicznego analizatora Knutha, Rado, Totha (1980), grupy pracowników Instytutu Automatyzacji Węgierskiej Akademii Nauk.

W powyższych przypadkach, brak jednak doniesień o praktycznych wynikach stosowania tych koncepcji na skalę przemysłową, jak to ma miejsce w przypadku ISDOS Project.

SYSTEM DEFINIOWANIA JĘZYKÓW OPISU SYSTEMU

Zbudowanie w oparciu o powyższy model poprawnego języka nie jest oczywiście zadaniem trywialnym, zwłaszcza jeżeli wyrażone w tym języku zdania mają poddawać się komputerowej analizie z punktu widzenia zamierzonych z góry funkcji. W ramach ISDOS Project opracowano w tym celu prototyp (rys. 2) programów wspomaganego komputerem systemu definiowania języków opisu systemów informacyjnych (Information Systems Language Definition System — ISLDS) [10]. Za pomocą tego narzędzia badacz-projektant bądź analityk, zamierzający poddać analizie określonego obiektu badań (bądź koncepcji projektowej) ze względu na określone aspekty, definiuje w dialogu z komputerem specyficzny do jego potrzeb język opisu systemów informacyjnych w oparciu o poprzednio wspomniany model obiekt — relacja — atrybut. Zdania definiujące obiekty, relacje i atrybuty — które mają występować w tym języku — są przez ISLDS przetwarzane, sprawdzane przy zakładaniu bazy danych oraz drukowaniu komunikatów diagnostycznych i ostrzeżeń pod względem jednoznaczności, syntaktycznej poprawności i wzajemnej zgodności z definicją języka. Użytkownikowi pozostawiono całkowitą swobodę w zakresie kolejności formułowania zdań definiujących język. Z chwilą, gdy autor języka jest już z jego budowy zadowolony, ISLDS może na żądanie użytkownika automatycznie wydrukować podręcznik hasłowy oraz ułożyć tablicę definiującą strukturę tego języka w formie przetwarzalnej przez komputer. Tablica ta będzie wykorzystywana do sterowania pracą programów analizujących zdania opisujące obiekt w tym języku.



Rys. 2. System definiowania języków opisu systemów informacyjnych — Information systems language definition system (ISLDS) wg Hershey (1980)

Zasadnicza różnica między podejściem do rozwiązania tego zagadnienia w ramach ISDOS Project a innymi podejściami (uprzednio omówionymi) polega na tym, że definiowanie języka opisu systemów informacyjnych za pomocą ISLDS jest etapem wyraźnie oddzielonym od użytkowania tego języka, a nie np. częścią procesu budowania bazy danych.

ISLDS został praktycznie wypróbowany i sprawdzony przy definjowaniu języków opisu systemów informacyjnych dla kilkunastu różnych technik.

Najszczegółowiej został opisany [23] eksperyment z Rational Design Methodology (RDM) z uwagi na szereg zalet tej techniki (sprawdzona poprawność, ograniczona złożoność, konstruktywne projektowanie, ściśle zdefiniowany język i kompletny opublikowany przykład zastosowania).

Natomiast największą natychmiastową przydatność praktyczną ma język opisu systemów (SDL), za pomocą którego staje się możliwe automatyczne wygenerowanie programu symulacyjnego w języku SIMSCRIPT II. 5, jako produktu ubocznego opisu w języku SDL badanego systemu informacyjnego (Spewak, 1980) [6]. Pozwala to na sprawdzenie dynamicznych aspektów funkcjonowania systemów, już nie tylko informacyjnych, lecz wszelkich systemów działania opisanych za pomocą tego języka. Jak wiadomo, wysiłek związany z tradycyjnym formułowaniem modelu symulacyjnego jest tak znaczny, że często powoduje rezygnację z symulacji lub przekreśla jej efekty.

Dla wymienionych technik każdy z języków zdefiniowanych za pomocą ISLDS jest szczególnym przypadkiem modelu obiekt — relacja — atrybut. Oczywiście różnią się one liczbą typów elementów włączonych do każdego języka, co ilustruje tabela. Jak łatwo zauważyć, w zestawieniu tym, najbardziej rozbudowanymi językami są PSL i SDL. Wiąże się to z zakresem stosowalności każdego języka.

Badania porównawcze zakresu stosowalności 12 różnych technik dla wspomagania projektowania systemów informacyjnych (przeprowadzone [22] gdy języka SDL jeszcze nie było) w pełni to potwierdzają. Chociaż żadne z porównywanych technik nawet swoim potencjalnym zakresem stosowalności (ze względu na 101 wyodrębnionych wówczas kryteriów) nie pokrywała całego badanego obszaru,

[5]) — Patrz relacja jako funktor zdaniotwórczy w logice [16]

[6]) — Podobne prace rozwojowe prowadzone są równolegle w Instytucie Informatyki Université N. D. de la Paix w Namur, Belgia (członek afillowany ISDOS Project) przez Bodart i Pigneur (1979), a ponadto jeszcze przez Alfroda (1977) i Willisa (1979). W tym ostatnim przypadku zresztą przy korzystaniu z tablicy sterującej symulatorem, który otrzymano z bazy danych PSL /PSA z ISDOS Project

to jednak już wówczas z uwagi na swą uniwersalność technika PSL/PSA [21] zdecydowanie wysuwała się na czoło, pokrywając faktycznie 60% a potencjalnie 74% badanego obszaru.

Tabela. Dane statystyczne o językach zdefiniowanych za pomocą ISDLS dla wybranych technik projektowania systemów informacyjnych wg Teichroew [23]

Technika	Typ obiektu	Typ relacji	Typ atrybutu	Typ tekstu	Liczba linii
Sadt	6	17	9	6	920
Composite design	17	25	3	2	1159
Data structure	19	19		3	1361
Real time	27	39	7	10	1479
RDM	19	20	4	15	1278
Process control	6	19	6	1	416
System matrix	5	20	11	3	1532
Software design	1	31	5	5	1208
Data definition	31	37	33	3	1609
PSL	17	66		10	1917
SDL	20	92	15/7	14	2413

Sprawą badacza, analityka, czy projektanta definiującego określony język opisu systemu jest rozstrzygnięcie, ile typów obiektów, relacji i atrybutów chce on wyróżnić ze względu na znane mu zamierzenia badawcze bądź rozwojowe oraz sformułowanie warunków i przypadków powodujących konieczność, możliwość lub niemożliwość ich wystąpienia. Będzie to również zależało od stopnia szczegółowości zamierzeń, zakresu sprawdzeń, analiz bądź planowanych eksperymentów symulacyjnych oraz stopnia ich automatyzacji. Większa liczba rodzajów obiektów, relacji i atrybutów umożliwia wprawdzie przeprowadzenie dokładniejszych analiz, ale z drugiej strony komplikuje język, opis systemu, zwiększa zapotrzebowanie na pojemność pamięci i moc obliczeniową komputera oraz wydłuża czas analizy.

Często projektant jest przyzwyczajony, bądź obowiązany do przestrzegania określonej procedury projektowania, reguł weryfikacji kompletności i poprawności rozwiązań oraz przygotowania dokumentów wynikowych o ściśle określonej treści i formie, np. treść Data Division dla programów pisanych w COBOL-u, schemat bazy danych w języku DDL, generacje modeli symulacyjnych w określonym języku symulacyjnym bądź programów źródłowych do kompilacji w określonym języku programowania. Projektant definiując język opisu systemu musi wówczas pamiętać, aby umożliwiał on spełnienie takich wymagań.

Cechą rozwiązań przyjętych w ISDOS Project jest umożliwienie całkowitej indywidualizacji metody i formy opisu oraz projektowania systemów informacyjnych. W porównaniu z innymi technikami, nie narzuca się użytkownikowi określonej metody postępowania lecz pozwala wybrać własną, otwiera możliwość korzystania prawie z każdej znanej metody oraz przygotowania narzędzi usprawniających posługiwanie się wybranymi metodami.

Te zalety sprawiają, że chociaż system definiowania języka opisu systemów ISLDS jest dopiero prototypem, ukazał się już komunikat o jego zastosowaniu w przemyśle. ISLDS użyto do opracowania języka RDL (Requirements and Development Language) według wymagań nie byle jakiego użytkownika, bo producenta komputerów SPERRY UNIVAC [8]. Język RDL jest wykorzystywany w pracach rozwojowych nad oprogramowaniem komputerów dla wewnętrznych potrzeb tej firmy.

LITERATURA

[1] Alford M.: A Requirements Engineering Methodology for Real-Time Processing Requirements. IEEE Transactions on Software Engineering, January 1977, Vol. SE-3, No 1, pp. 60—69

[2] Bodart F., Pigneur Y.: A Model and Language for Functional Specification and Evaluation of Information System Dynamics. Proceedings of the IFIP TC8 Working Conference on Formal Methods and Practical Tools for Information System Design, Oxford 1979, North-Holland

[3] Boyd D., Pizzarello A., Vestal S.: Rational Design Methodology. Report HR-78-257: 17—38, Honeywell INC., Rome Air Development Center, Information Sciences Division, Griffiss AFB, NY 13441, June 1978, 178 pp. (ADA06 2404)

[4] Chen P.: The Entity — Relationship Model — Toward a Unified View of Data. ACM Transactions on Database Systems, Vol. 1, No 1, March 1976, pp. 9—36

[5] Davenport R.: Data Analysis for Database Design. The Australian Computer Journal, Vol. 10, No 4, November 1978, pp. 122—137

[6] Gackowski Z.: Próba operacyjnej formalizacji ogólnego pojęcia systemu. Problemy Organizacji z. 1/1979, s. 43

[7] Hammer M., Mc Loed D.: The Semantic Data Model: A Modelling Mechanism for Data Base Application. Proceedings, ACM SIGMOD 1978, pp. 25—36

[8] Heacox H.: RDL: A Language for Software Development. ACM SIGPLAN Notices, Vol. 14, No 12, December 1979, pp. 71—79

[9] Hoar, C. A. R.: Characteristics Needed for a Common High Order Programming Language. Written Comments to ISDOS Project, 1975

[10] Hershey E A. III: A system for Defining Information System Description Languages. Ph. D. Disseration. The University of Michigan, 1980

[11] ISDOS Ref // 79A42-0249-0- Problem Statement Language Reference, Summary, 1979

[12] Jones M. N.: HIPO for Developing Specifications. Datamation, March 1976, pp. 112—125

[13] Jefferson D. K., Taylor D. W.: Analysis of Requirements for Large-Scale Information Systems. Proceedings IEEE Compac 79, Chicago, November 5—8, 1979

[14] Knuth E., Rado P., Tóth A.: Preliminary Description of SDLA/Structural Descriptor and Logical Analyzer. Institute of Automation, Hungarian Academy of Sciences, December 1979, Tanulmányok 1980

[15] McLeod D.: A Semantic Data Base Model and Its Associated Structured User Interface. Massachussetts Institute of Technology, Ph. D. Thesis, August, 1978

[16] MEL — Mała Encyklopedia Logiki. Zakład Narodowy im. Ossolińskich, Wrocław, 1970

[17] Nijssen G., Van Assche F., Snijders J.: End — User Tools for the Most Abstract Information Systems Requirements Definition. IFIP WG 8.1 Working Conference, Oxford, England, April 17—20, 1979

[18] Spewak S.: Analysis of Dynamic of the Logical Design of Information Systems, Ph D. Disseration. The University of Michigan (expected 1980)

[19] Stavely A: A Design Feedback and Its Use in Software Design Aid Systems. Proceedings of the Software Quality and Assurance Workshop, ACM, November 1979, San Diego, CA, pp. 72—78

[20] Teichroew D.: A Survey of Languages for Stating Requirements for Computer — Based Information Systems. Fall Joint Computer Conference Proceedings, Vol. 41, part II/Montvale, New Jersey, AFPS Press, 1972

[21] Teichroew D., Hershey E. A. III: PSL/PSA: A Computer — Aided Technique for Structured Documentation and Analysis of Information Proceesing Systems. IEEE Transaction of Software Engineering, February 1977

[22] Teichroew D., Gackowski Z.: Development and Documentation Methods of Information Proceesing Systems. ISDOS Research Project. The University of Michigan, Ann Arbor, Michigan, May, 1977

[23] Teichroew D., Macasovic P., Hershey E. A. III., Yamamoto Y.: Application of the Entity-Relationship Approach to Information Proceesing System Modelling. ISDOS Project, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan 48109, October 1979. Paper for the International Conference on Entity-Relationship Approach to Systems Analysis and Design, December 10—12, 1979, Los Angeles

[24] Willis R.: DAS An Automated System to Support Design Analysis. Proceedings of the 3 rd International Conference on Software Engineering, May 1978, pp. 109—115

[25] Winters E. W.: Experience with Problem Statement Language: A Tool for. Structured Documentation. Proceedings, Application Development Symposium, Sponsored by GUIDE. International, Inc., SHARE, Inc., and IBM, Montrey, CA, October 14—17, 1979, pp. 137—141.

W pierwszej części artykułu Autorzy przedstawili strukturę LINDY, systemu służącego do budowy systemów informacyjnych z bazą danych średniej wielkości, oraz ogólną jej charakterystykę. Część druga omawia pozostałe, szczegółowe zagadnienia, a przede wszystkim — cechy oraz możliwości zastosowań języka manipulacji danymi, wykorzystywanego w LINDZIE. (Red.)

WIKTOR RZECZKOWSKI, KAZIMIERZ SUBIETA
Instytut Podstaw Informatyki PAN
Warszawa

LINDA — system zarządzania bazą danych. Część 2

W języku manipulacji danymi (JMD) wyrażenia języka identyfikacji wykorzystywane są w następujących celach:
identyfikatory — dla określenia miejsca (adresu) przechowywania danych
termy — dla określenia wartości
formuły — dla określenia warunku.

Język manipulacji danymi systemu LINDA obejmuje następujące grupy instrukcji:

- instrukcje dotyczące druku lub transmisji do taśmowych zbiorów listowania:

#CHARS — drukowanie poziomych linii
#TAB — tabulacja wydruków
#LINE — zmiana wiersza
#PAGE — zmiana strony
#PRINT — drukowanie tekstów (wartości termów)
#VLINE — drukowanie pionowych linii
#FORMAT — ustalenie postaci znakowej drukowanych liczb
#LISTFILE — ustalenie nośnika, na który będą wyprowadzane informacje w wyniku instrukcji druku; może to być drukarka lub taśma magnetyczna
#CLOSE — zamykanie taśmowego zbioru listowania;

- instrukcje podstawienia:

#ASSIGN — podstawienie nowej wartości do określonej danej
#ADD — dodanie wartości do wartości określonej danej

- instrukcje wzbogacania pola widzenia (części bazy danych, na której operuje identyfikator) o nowe dane oraz usuwania z niego danych:

#CREATE — wytworzenie danej złożonej
#ASSIGN — wytworzenie danej elementarnej
#DELETE — usunięcie danej

- instrukcje odwołujące się do bazy danych na nośnikach zewnętrznych:

#GET — przekazanie kolejnych rekordów z nośnika zewnętrznego do pola widzenia
#STORE — przekazanie rekordów z pola widzenia na nośnik zewnętrzny

- instrukcja odwołania się do modułu zarządzającego:

#CONTROL — przy jej użyciu można wydać dowolną instrukcję modułu zarządzającego w trakcie wykonywania programu manipulacyjnego

- instrukcja skoku bezwarunkowego:

#GOTO — do miejsca w programie oznaczonego wskazaną etykietą

- instrukcje związane z czytaniem kart dziurkowanych:

#RMODE — ustalenie trybu czytania, tj. znaków kończących wczytywaną jednostkę danych i (lub) zalecenie przetwarzania liczb na postać binarną
#READ — czytanie tekstu z karty
#NEWCARD — zmiana karty

- instrukcje warunkowe i strukturalne:

#IF — warunkowa
#CASE — strukturalna, umożliwiająca dowolne przełączenie procesu obliczeniowego
#FOR — umożliwiająca sekwencyjne przetwarzanie wielowartościowych termów
#WHILE, #REPEAT, #CYCLE — instrukcje inicjujące pętle programowe;

- instrukcja odwołania do procedury:

CALL — w programach manipulacyjnych można używać procedur bezparametrowych, rekursywnych. W procedurach nie można deklarować wielkości lokalnych;

- instrukcja pusta:

#NULL — umożliwia m. in. zaetykietowanie dowolnego miejsca w programie manipulacyjnym (np. końca programu). Dowolna instrukcja języka manipulacji może być poprzedzona etykietą.

Dla przykładu podamy postać gramatyczną i opis semantyczny kilku wybranych instrukcji JMD (symbole metajęzykowe są podkreślone):

#ASSIGN<literal>, <term>;
Instrukcja ta powoduje wytworzenie danych o nazwie zadanej literalem i wartościach wyznaczonych przez term. Danych tych utworzy się tyle, ile będzie wartości termu.
#ASSIGN<identyfikator>, <term>;
Instrukcja podstawienia; pod kolejny adres wyznaczony identyfikatorem podstawia się kolejną wartość termu. W przypadku niezgodności liczby znaczeń identyfikatora i termu następują odpowiednie sygnalizacje.

#CREATE<definicja rekordu>;
<definicja rekordu> : : = <literal> : (<term>)
<definicja rekordu> : : = <literal>(<definicja rekoru> {<definicja rekordu>})
Instrukcja powoduje wytworzenie rekordu o dowolnej strukturze hierarchicznej. Wartości elementarne w rekordzie zadaje się za pomocą termów, a nazwy danych tworzących rekord — za pomocą literali.

#GET[<identyfikator>] <ciąg instrukcji>[#UNTIL <formuła>]#TEG
Instrukcja powoduje wykonanie ciągu instrukcji dla kolejnych rekordów ze zbioru wejściowego lub wykonanie ciągu instrukcji kolejno dla wszystkich rekordów wskazanych identyfikatorem, jeżeli identyfikator występuje. Użycie opcyjnej frazy #UNTIL<formuła> powoduje zakończenie instrukcji w chwili, gdy formuła stanie się prawdziwa, co umożliwia przerwanie cyklu wykonywania ciągu instrukcji przed osiągnięciem końca zbioru.
#FOR<literal>#IN<term># DO <ciąg instrukcji> #FI
Instrukcja wytwarza dane o nazwie określonej literalem i wartości określonej kolejną wartością termu. Dla każdej takiej danej wykonywany jest ciąg instrukcji.

Dla ułatwienia czynności związanych z przygotowaniem, przechowywaniem, uruchamianiem i wykorzystaniem programów manipulacyjnych wprowadzono następujące możliwości:

- programy manipulacyjne mogą być wstępnie zapisane na taśmę magnetyczną przy użyciu edytora #XKYA, dzięki czemu jest możliwe ich proste poprawienie i przechowywanie
- w czasie kompilacji programu manipulacyjnego można wyprowadzić tekst programu z ewentualnymi sygnalizacjami błędów i rozbiorem leksykalnym tekstu

- w programach manipulacyjnych można zadeklarować fragmenty (np. procedury), które będą dołączane z taśmy magnetycznej (tzw. fragmenty biblioteczne)
- w programach manipulacyjnych znajdujących się na taśmach magnetycznych można zadeklarować fragmenty, które podczas kompilacji zostaną przeczytane z kart dziurkowanych (tzw. makrosy)
- kompilację programu manipulacyjnego można wykonać w specjalnym trybie (ERTRACE), w którym do wynikowego kodu programu zostaną dołączone fragmenty diagnostyczne; dzięki temu przy wykonaniu programu manipulacyjnego można wydrukować przejścia przez etykiety programowe, niektóre instrukcje (takie jak:#GET, #CASE, #IF), wejścia do procedur oraz wartości boolowskie niektórych formuł; fragmenty diagnostyczne umożliwiają także odpowiednie sygnalizacje i (lub) zatrzymania w razie wystąpienia — w trakcie wykonywania programu — sytuacji awaryjnych lub niepewnych.

PRZYKŁAD UŻYCIA JMD

Dla ilustracji możliwości języka manipulacji danymi SZBD LINDA przedstawimy przykładowy program manipulacyjny, który ma znaczenie wyłącznie dydaktyczne.

Załóżmy, że zbiór dyskowy BAZA DANYCH zawiera rekordy o nazwie PRACOWNIK (o strukturze opisanej wyrażeniem języka opisu danych z części 1). Rekordy dotyczą kobiet i mężczyzn zatrudnionych w pewnym przedsiębiorstwie. Chcemy obliczyć ile kobiet i ilu mężczyzn w podeszłym wieku pracuje w tym przedsiębiorstwie oraz przesłać do istniejącego zbioru dyskowego STARSI PRAC rekordy wszystkich pracowników w podeszyłym wieku, zmieniając jednocześnie nazwę tych rekordów z PRACOWNIK na STARSZY PRACOWNIK. Zakładamy, że pod pojęciami kobieta czy mężczyzna w podeszłym wieku rozumiemy osoby w wieku odpowiednio powyżej 50 i 55 lat. Wymieniony cel można osiągnąć za pomocą programu manipulacyjnego przedstawionego poniżej.

```
# PROGRAM PRZYKLAD;
# CONTROL INEDS (BAZA DANYCH), OUTEDS (STARSI
  PRAC),
# EXIT, # ASSIGN A, ϕ; # ASSIGN B, ϕ;
# GET [PRACOWNIK:
  [PLEC] = K # AND # DATE — [DATA URODZENIA] >
  # DATINT 50 # OR
  [PLEC] = M # AND # DATE — [DATA URODZENIA] >
  # DATINT 55]
# IF [PRACOWNIK; PLEC] = K # THEN # ADD [A], 1;
# ELSE # ADD [B], 1; # FI
# CREATE STARSZY PRACOWNIK: ([PRACOWNIK]);
# STORE [STARSZY PRACOWNIK];
# DELETE [STARSZY PRACOWNIK];
# TEG
# CONTROL INCL, OUTCL, # EXIT,
# PRINT ↑', 'STARSI PRACOWNICY', ↑,'==========
  ======,↑',
        ↑, 'LICZBA STARSZYCH KOBIET =', [A],
        ↑, 'LICZBA STARSZYCH MĘŻCZYZN =,' [B], ↑;
# CHARS ' * ', 35;
# END
```

Na wstępie program otwiera dwa zbiory dyskowe: wejściowy do odczytu BAZA DANYCH i wyjściowy do zapisu STARSI PRAC (instrukcja #CONTROL). Z kolei tworzone są dwie robocze dane elementarne A i B z początkowymi wartościami: O (#ASSIGN). Następna instrukcja #GET powoduje przegląd zbioru wejściowego z bazą danych. Dla każdego rekordu pracownika dostarczonego do pola widzenia, takiego że spełniony jest jeden z warunków:

- rekord opisuje kobietę (pole PLEC ma wartość K) oraz wiek pracownika (tj. różnica dni między datą bieżącą a datą w polu DATA URODZENIA) jest większy niż 50 lat.
- rekord opisuje mężczyznę (pole PLEC ma wartość M) oraz wiek pracownika jest większy niż 55 lat

wykonuje się instrukcje zawarte w instrukcji #GET, tj. ciąg instrukcji, #IF, #CREATE, #STORE i #DELETE.

Jeśli aktualnie dostarczony rekord dotyczy kobiety (porównaj formułę po #IF), to wartość danej A powiększa się o 1, w przeciwnym razie — wartość danej B powiększa się o 1. Kolejna instrukcja ciągu (#CREATE) tworzy nowy rekord o nazwie STARSZY PRACOWNIK, o strukturze i wartościach pól takich samych, jak w dostarczonym aktualnie rekordzie PRACOWNIK. Nowo utworzony rekord jest przesyłany do zbioru wyjściowego (#STORE), a następnie usuwany z pola widzenia (#DELETE). Po zakończeniu przeglądu bazy danych (instrukcja #GET) następuje zamknięcie zbiorów wejściowych i wyjściowego (#CONTROL) i na drukarce zostaje wypisana pożądana informacja. Wydruk realizują dwie instrukcje: #PRINT i #CHARS. Instrukcja #PRINT drukuje określony w niej ciąg wartości oddzielonych przecinkami. Znak strzałki (↑) oznacza zmianę wiersza. Instrukcja #CHARS podkreśla wydruk ciągiem 35 znaków gwiazdki. Jeśli na zakończenie przeglądu bazy danych zmienne A i B przyjęły wartości odpowiednio 50 i 65, to program dostarczy wydruk o następującej postaci:

```
STARSI PRACOWNICY
==========
LICZBA STARSZYCH KOBIET      = 50
LICZBA STARSZYCH MĘŻCZYZN = 65
**********
```

OBLICZANIE STATYSTYK

Mechanizmy obliczania statystyk zostały zrealizowane jako instrukcje języka manipulacji danymi. W rezultacie działania tych instrukcji obliczane są statystyki dotyczące występowania określonej cechy (lub pary cech) w zbiorze danych. Wynikiem działania instrukcji statystycznych jest utworzenie w pamięci operacyjnej nowych danych, zawierających odpowiednie informacje statystyczne. Dane te mogą być w dalszym ciągu przetwarzane przy użyciu dowolnych instrukcji języka manipulacji.

Opracowano następujące instrukcje obliczania statystyk:

FREQCALC — obliczanie dla próby rozpatrywanej ze względu na pojedynczą cechę X: w przypadku cechy X o wartościach dyskretnych — funkcji częstości, natomiast w przypadku cechy X o wartościach ciągłych (liczbowych) — funkcji gęstości oraz wartości średniej arytmetycznej i odchylenia standardowego

AVERAGE — obliczanie średniej arytmetycznej z próby (tj. estymatora wartości oczekiwanej) dla pojedynczej cechy X o wartościach liczbowych

STANDEV — obliczanie odchylenia standardowego z próby od zadanej wartości oczekiwanej dla pojedynczej cechy X o wartościach liczbowych

XYSTATIS — obliczanie dla próby rozpatrywanej ze względu na dwie cechy X i Y; w przypadku cech X i Y o wartościach dyskretnych — funkcji częstości dla pary cech X i Y, natomiast w przypadku gdy cechy te mają wartości liczbowe — funkcji gęstości dla pary cech, wartości oczekiwanych, odchyleń standardowych dla każdej z cech oraz współczynnika korelacji między cechami X i Y.

Dzięki wprowadzeniu do języka manipulacji instrukcji statystycznych, użytkownik uzyskuje środek dla łatwego formułowania programów statystycznego przetwarzania zbiorów danych. Jednocześnie, z uwagi na to, że procedury wykonawcze odpowiadające tym instrukcjom zostały napisane w języku PLAN (tj. w języku asemblera komputera ODRA) — czas wykonania zadań statystycznych sformułowanych z użyciem instrukcji statystycznych jest o rząd mniejszy niż czas wykonania równoważnych im programów manipulacyjnych, nie zawierających instrukcji wyznaczania statystyk.

AKTUALIZACJA DANYCH

Przy użyciu języka manipulacji danymi systemu LINDA można przeprowadzić aktualizację danych, ale w ograniczonym zakresie (brak możliwości dostawiania i usuwania danych w rekordach, możliwa jest wyłącznie zmiana wartości istniejących danych). Szersze możliwości są zawarte w specjalnym języku aktualizacji danych, który w wygodny sposób pozwala wyrazić wszelkie zlecenia dotyczące zmian w bazie danych.

Poprawki można wykonywać w jednym z następujących trzech trybów:

FIREMOVE — kolejność poprawek jest taka sama jak kolejność rekordów w zbiorze, każda poprawka dotyczy jednego rekordu

ALREMOVE — kolejność poprawek jest dowolna, każda poprawka dotyczy jednego rekordu (tryb bardziej pracochłonny)

ALLEAVE — kolejność poprawek jest dowolna, każda poprawka może dotyczyć wielu rekordów; ten tryb może być użyty do globalnych zmian w bazie danych.

Język aktualizacji danych ma dwa poziomy. Na pierwszym poziomie instrukcje aktualizacyjne dotyczą rekordu. Instrukcje te są następujące:
DELETE — usuń określony rekord
INSERT # AFTER — wstaw nowe rekordy za określony rekord
INSERT #BEFORE — wstaw nowe rekordy przed określony rekord
INSERT #AFTER #LAST — wstaw nowe rekordy na koniec zbioru
MODIFY — zmień zawartość określonego rekordu.

W instrukcjach typu #INSERT w charakterze parametrów występują nowe rekordy. Rekordy te mogą być podane w stałym lub w zmiennym formacie, tak jak w programach wprowadzających dane. W trakcie wczytywania poprawek może być skontrolowana zgodność tych rekordów z wyrażeniem JOD.

Na drugim poziomie języka aktualizacji znajdują się instrukcje używane wewnątrz instrukcji #MODIFY, za pomocą których można zmienić zawartość rekordu w dowolny sposób. Ponieważ wszystkie rekordy fizyczne mają zmienny format, zatem jest możliwe dokonywanie dowolnych wstawek do rekordu, rozszerzeń, usunięć i zmniejszania obszaru zajmowanego przez daną wartość, przy czym zmiany tę są dokonywane na poziomie fizycznym.

W rekordach poprawionych, podobnie jak w rekordach fizycznych, utworzonych w procesie wprowadzania danych, dane są ulokowane ściśle (bez luk). Instrukcje umożliwiające realizację wymienionych rodzajów zmian w rekordach są następujące:
DELETE — usunięcie danych (pól, podpól, itd.) z rekordu
REPLACE — zmiana danej w rekordzie na nową
INSERT #BEFORE — wstawienie nowej danej przed określonym miejscem
INSERT #AFTER — wstawienie nowej danej za określonym miejscem
RENAME — zmiana nazwy rekordu lub dowolnej zawartej w nim danej
ASSIGN — podstawienie nowej wartości do określonej danej w rekordzie.

Miejsce w rekordzie, w którym ma być dokonana zmiana jest ustalane za pomocą identyfikatora. Ponadto miejsce to można określić używając kwalifikatorów „nawigacyjnych", takich jak:
FIRST — pierwsza dana spośród wskazanych
LAST — ostatnia dana spośród wskazanych
NEXT — dana następna po wskazanych
PRIOR — dana poprzedzająca dane wskazane
BELOW — dane podrzędne w stosunku do wskazanych
ABOVE — dane nadrzędne w stosunku do wskazanych.

FUNKCJE SPECJALNE

System LINDA daje możliwość wykonania szeregu funkcji specjalnych, które ułatwiają sprawną jego eksploatację. Funkcje te są następujące:
- możliwość automatycznego stronicowania wydruków, drukowanie nagłówków stron na wydrukach pochodzących od dowolnych podprogramów użytkowych systemu i programów manipulacyjnych
- automatyczne zakładanie zbiorów taśmowych i dyskowych
- drukowanie zbiorów danych (rekordów) w postaci standardowej (DISPLAY)
- kopiowanie zbiorów (taśmowych lub dyskowych) lub ich części na wskazany nośnik, scalanie zbiorów taśmowych lub dyskowych (REWRITE)
- wyszukiwanie rekordów spełniających zadane kryterium logiczne (SELECT)
- sortowanie zbiorów dyskowych według dowolnego kryterium (SORT)
- drukowanie kart perforowanych, zbiorów zapisanych za pomocą edytora #XKYA lub taśmy błędów (PRINT)
- drukowanie zbiorów taśmowych w formacie drukarki wierszowej, wytworzonych w programie manipulacyjnym (LISTFILE).

W przygotowaniu znajdują się rozszerzenia systemu, umożliwiające generację i wykorzystanie kartotek inwersyjnych.

* * *

Doświadczenia z eksploatacji systemu LINDA wykazały, że jest on wygodnym narzędziem do budowy systemów informacyjnych z bazą danych średniej wielkości, w różnych dziedzinach zastosowań. Zaletą podkreślaną przez użytkowników jest wielokrotne skrócenie czasu niezbędnego do zaprojektowania, zrealizowania i wdrożenia systemu informacyjnego, dzięki czemu zamierzenia — dotychczas nieosiągalne — stają się realne i opłacalne. Efekt ten osiągnięto poprzez wyposażenie systemu w elastyczne i uniwersalne środki językowe, pozwalające na wygodne, niezawodne, a przy tym odpowiednio efektywne przetwarzanie danych. W odróżnieniu od większości znanych obecnie systemów zarządzania bazą danych, system LINDA charakteryzuje się tym, że bardzo łatwo i szybko można nauczyć się nim posługiwać. Przeciętnie zorientowanemu w zagadnieniach informatyki użytkownikowi dla zdobycia umiejętności korzystania z SZBD LINDA wystarcza krótki (np. tygodniowy) okres zapoznawania się z systemem.

System LINDA jest udostępniany przez Instytut Podstaw Informatyki PAN, skr. poczt. 22, 00-901 Warszawa PKiN.

WCCE 81 — III światowa konferencja „Komputery w nauczaniu"

Kolejna, trzecia już, światowa konferencja IFIP na temat komputerów w nauczaniu (WCCE — World Conference on Computers in Education) odbędzie się w dniach od 27 do 31 lipca 1981 r. w Lozannie (Szwajcaria). Jej organizatorem jest z upoważnienia IFIP Szwajcarska Federacja Automatycznego Sterowania, natomiast patronat nad nią objęły UNESCO oraz jedna z jego agend — Międzyrządowe Biuro Informatyki (IBI — Intergovernmental Bureau for Informatics) w Rzymie, które subsydiować będą delegatów z krajów rozwijających się (w pierwszym rzędzie autorów referatów). Spodziewany jest udział ponad 1000 uczestników (dotychczas zgłosiło się 1157 osób, w tym ok. 70 z krajów RWPG).

Głównym tematem konferencji będą zastosowania komputerów na wszystkich szczeblach kształcenia (od szkół podstawowych po szkoły wyższe) oraz w szkoleniu zawodowym. Uzupełnieniem konferencji będą krajowe turnieje młodzieży w programowaniu, których laureaci zostaną zaproszeni do zaprezentowania na konferencji swych osiągnięć. W czasie trwania konferencji zorganizowana zostanie również wystawa obrazująca obecny stan materiałów dydaktycznych oraz reprezentatywne specjalistyczne rozwiązania sprzętowe i oprogramowanie.

W referatach podkreślone zostaną takie tematy, jak stosunek informatyki do nauczania innych przedmiotów, metody nauczania wspomaganego komputerem (CAL — Computer Aided Learning), wpływ informatyki na nauczycieli i studentów oraz wykorzystanie wolnego czasu, a także narodowe modele i programy komputeryzacji nauczania, ze szczególnym uwzględnieniem potrzeb i możliwości krajów rozwijających się.

W planowanych podczas konferencji dyskusjach panelowych zaproponowano uzyskanie odpowiedzi na następujące aktualne pytania:
- co każdy inteligentny człowiek powinien wiedzieć o informatyce?
- jak kształcić informatyków aby byli oni „społecznie odpowiedzialni"?
- czy szkoły średnie należy wyposażać w komputer centralny z terminalami, czy też raczej w mikrokomputery?
- jakie nowe osiągnięcia techniczne będą miały istotny wpływ na metody kształcenia w ciągu najbliższych 5 lat?

Na marginesie powyższej zapowiedzi należy odnotować rosnące opłaty uczestnictwa, które wynoszą tu 300 franków szwajcarskich (ok. 180 dol. USA), chociaż w przypadku wcześniejszych wpłat podlegają one znacznym bonifikatom (200 fr. przy wpłatach do 31.1.81 oraz 250 fr. — do 31.5.81).

W ciągu kilku ostatnich lat powstało wiele modeli umożliwiających szacowanie niezawodności programów. Modele te określają przybliżoną liczbę błędów oraz czas i wielkość personelu, potrzebnego do ich usunięcia. W większości przypadków były one oparte na rzadko spełnionych założeniach, wskutek czego niezbyt często wykorzystywano je praktycznie. Poza tym wiele z nich należy traktować jako teoretyczne hipotezy. Tym niemniej niektóre zostały zweryfikowane, przynajmniej dla pewnych zastosowań. Wyróżnić tu należy model Schicka-Wolvertona, Ottenstein oraz Musy. Zwłaszcza ten ostatni, będący pewną syntezą wcześniej utworzonych modeli, znalazł stosunkowo najszersze zastosowanie (w szczególności w USA i Francji). Według informacji posiadanych przez autora modele te nie są jeszcze w Polsce wykorzystywane, przede wszystkim ze względu na brak odpowiedniej statystyki błędów, chociaż niektóre (np. model Ottenstein, najprostszy i dość dokładny) warte są szerszego wykorzystania.

WITOLD KORNACKI
Instytut Informatyki
Politechnika Warszawska

Wybrane modele niezawodnościowe programów

W poprzednim numerze INFORMATYKI został opublikowany artykuł nt. analizy niezawodnościowej oprogramowania, będący wprowadzeniem do tematyki niniejszego opracowania, w którym przedstawiono charakterystyczne rodzaje modeli niezawodnościowych programu, zgrupowanych w dwie zasadnicze klasy:
- modele w domenie czasu
- modele w domenie danych.

Pierwsze z nich oparte są na rozkładach (ciągłych i dyskretnych) prawdopodobieństwa błędów programowych. Przy znajomości rozkładu występowania błędów w funkcji czasu testowania lub czasu użytkowania programów, modele te umożliwiają określenie szeregu parametrów niezawodnościowych.

W modelach drugiej klasy zakłada się znajomość wartości wszystkich zmiennych wejściowych lub przynajmniej wartości reprezentatywnych dla określonego zastosowania programu. Wykonując (testując) program na tych zbiorach danych można oszacować jego niezawodność.

Modele w domenie czasu umożliwiają znacznie szerszą analizę niezawodnościową oprogramowania. W praktyce zaś, oba typy modeli uzupełniają się wzajemnie, czyniąc analizę niezawodnościową bardziej wszechstronną.

MODELE NIEZAWODNOŚCIOWE W DOMENIE CZASU

W modelach z tej klasy wykorzystuje się klasyczną definicję niezawodności R(t) [3], rozumianej jako prawdopodobieństwo bezbłędnego (zgodnego ze specyfikacjami) wykonywania programu w okresie (o, t). W tym przypadku:

$$R(t) = \exp\left(-\int_0^t \lambda(x)\,dx\right)$$

Jeśli program jest poza fazą testowania, często przyjmuje się stałą wartość współczynnika intensywności błędów programu $\lambda(t) = \lambda = const$, co prowadzi do modelu wykładniczego [10, 13]. Do oszacowania λ służą znane metody [10]. Wagoner w pracy [15] przyjął następującą postać $\lambda(t)$:

$$\lambda(t) = \alpha \cdot \beta \cdot t^{\beta-1}, \quad t > o; \quad \alpha, \beta > 0$$

gdzie α, β — stałe wyznaczane statystycznie

co prowadzi do określania $R(t)$ przez dwuparametryczny rozkład Weibulla. Ponieważ $\lambda(t)$ jest najczęściej funkcją malejącą, więc przyjmuje się $\beta < 1$. Model ten wykazał dużą zgodność z danymi zebranymi w toku eksploatacji kilkunastu dużych programów [13].

W roku 1972 Jelinski i Moranda wprowadzili w pracy [5] dyskretnie malejącą, przedziałami stałą funkcję intensywności błędów w postaci

$$\lambda = \Phi(N - i)$$

gdzie

N — początkowa liczba błędów w programie
i — liczba wykrytych błędów
Φ — stała proporcjonalności

Przyjęli oni, że współczynnik λ jest proporcjonalny (ze stałą Φ) do liczby błędów w programie. Ponadto założyli, że każdorazowo w procesie korekcji programu usuwa się dokładnie jeden błąd (założenie często niespełnione). Opierając się na powyższej zależności otrzymali estymatory największej wiarygodności parametrów φ i N.

W połowie tego samego roku Schick i Wolverton [12] podali swój pierwszy model niezawodnościowy, w którym przyjęli liniową zależność pomiędzy dyskretnie malejącą funkcją ryzyka a czasem testowania:

$$\lambda(\tau_i) = \Phi(N - i)\,\tau_i$$

gdzie

τ_i — długość czasu testowania do wystąpienia i-tego błędu. Podali również postać estymatorów Φ i N. Dodajmy, że w celu ich oszacowania należy dysponować odpowiednią statystyką błędów. Schick i Wolverton zaproponowali też pewną modyfikację swego modelu [13], przyjmując nieliniową (opisaną wielomianem kwadratowym) zależność $\lambda(\tau)$ od czasu testowania.

Shooman [14] przyjął w swym modelu, że stosunek liczby błędów w programie do liczby instrukcji ma wartość ustaloną, równą tzw. **stałej błędów kodowania** (ang. *coding-error rate*), określanej w przybliżeniu na 2 błędy do 100 instrukcji. Założenie to było niejednokrotnie krytykowane i — jak wskazują dane zawarte w tabeli 1 (otrzymane na podstawie obserwacji 10 dużych programów [13]) — wydaje się prowadzić do zbyt dużych rozbieżności.

Funkcja niezawodności podana przez Shoomana ma postać następującą:

$$R(t) = \exp(-\lambda(\tau) \cdot t)$$

gdzie:

$\lambda(\tau)$ — współczynnik intensywności błędów programu po czasie testowania τ

$$\lambda(\tau) = C\left(\frac{E}{I} - \varepsilon(\tau)\right)$$

zaś

E — liczba błędów w programie (określona na podstawie stałej błędów kodowania)
I — liczba instrukcji programu
$\varepsilon(\tau)$ — liczba usuniętych błędów programowych po czasie τ
C — stała [6].

Tabela 1. Wartości stałej błędów kodowania dla różnych faz tworzenia programu

Fazy tworzenia programu	Błędy/100 instrukcji
A	0.786
B	0,504
C	0,371
D	0,337
x	0,495
σ(x)	0,196

Oznaczenia;

A, B, C, D — kolejne fazy tworzenia programu
x — wartość średnia stałej błędów kodowania
σ(x) — odchylenie standardowe od wartości średniej

W ciągu ostatnich kilku lat czynione były próby powiązania złożoności programu z jego niezawodnością [2]. Intuicja podpowiada, że większej złożoności programu odpowiada mniejsza niezawodność. Zostało to potwierdzone przez L. M. Ottenstein w roku 1979 [11] na drodze analizy statystycznej szeregu programów. Zaobserwowała ona również, że w większych programach występuje mniejsza stopa błędów. W interesujący sposób powiązała liczbę błędów w programie z pewnymi miarami złożoności programu, z uwzględnieniem szeregu czynników natury psychologicznej.

Liczbę błędów N w programie można ocenić według jej metody z zależności:

$$N = \frac{I_V \log_2 \eta}{3000}$$

gdzie:

I_V — liczba operatorów i operandów użyta w programie o długości I instrukcji
Wartość η liczy się najczęściej z zależności:

$$I_V \approx \begin{cases} \frac{8}{3} \cdot I \text{ dla języka typu asembler} \\ \frac{15}{2} \cdot I \text{ dla języków wyższego rzędu} \end{cases}$$

η — liczba operatorów i operandów występujących w programie **jednokrotnie.**
Wartość η liczy się najczęściej w zależności:

$$I_V = \eta \log_2 (\eta/2)$$

Model ten, przedstawiony w znacznym uproszczeniu, odznacza się stosunkowo dużą dokładnością (±50% przewidywanej liczby błędów). W oparciu o istniejące już modele można na jego podstawie wyznaczyć szereg parametrów niezawodnościowych. Model ten może być jednak zbyt niedokładny dla programów małych.

Goel w pracy [4] uzyskał, przy wykorzystaniu teorii procesów półmarkowskich przybliżoną zależność na niezawodność programu $R_k(t)$ po wykryciu k błędów, dopuszczając możliwość usuwania błędów z prawdopodobieństwem różnym od 1. Zwiększył w ten sposób możliwości aplikacyjne modelu.

W normie branżowej [10] (z 1978 r.) podano metodę wyznaczania współczynnika intensywności błędów programu λ na podstawie zarejestrowanej statystyki błędów. Przyjęto, że w okresie użytkowania programu ma on wartość stałą. Podano również zależność na oszacowanie parametru nazwanego **współczynnikiem jakości oprogramowania,** określanego jako stosunek czasu poprawnego wykonywania programu do łącznego czasu jego wykonywania oraz czasu traconego w wyniku błędów. Dodajmy, że zgodnie z normą wartość tego współczynnika nie może być mniejsza od 0,95.

Przedstawione dotychczas modele pierwszej klasy są jej typowymi reprezentantami. Istnieją jeszcze modele oparte na hipergeometrycznym rozkładzie błędów w programie [1, 13], ale ponieważ nie zapewniają one wyznaczenia parametrów niezawodnościowych (wymagają w tym celu posłużenia się dodatkowo innymi modelami) nie będą tu omawiane.

MODELE NIEZAWODNOŚCIOWE W DOMENIE DANYCH

Traktując program π jako pewne odwzorowanie zbioru danych wejściowych I na zbiór danych wyjściowych O:

$$\pi : I \to 0$$

możemy określić zbiór F takich wartości danych wejściowych, które powodują wystąpienie defektu wykonania

$$F = \{i \in I : \pi(i) = \overline{\pi(i)}\}$$

gdzie:

$\pi(i)$ — dana wyjściowa
$\overline{\pi(i)}$ — dana wyjściowa niezgodna ze specyfikacją

Niezawodność oprogramowania określa się w tym przypadku [13] jako:

$$\hat{R} = 1 - \frac{card\,(F)}{card\,(I)}$$

$\hat{R}$ można interpretować jako prawdopodobieństwo bezbłędnego jednorazowego wykonania programu (z danymi wejściowymi wybieranymi losowo ze zbioru I).

Określenie zbioru F stwarza olbrzymie trudności. Jeśli uwzględnimy fakt, że wartości poszczególnych zmiennych są wzajemnie zależne, to definicja powyższa przestaje być adekwatna. W ogólnym przypadku jest ona słuszna tylko dla jednej zmiennej wejściowej.

Praktycznie $\hat{R}$ estymuje się [13] za pomocą zależności:

$$\hat{R} = 1 - \frac{n_e}{N}$$

gdzie:

n_e — liczba poprawnych przebiegów programu z danymi wybieranymi losowo (albo według określonego rozkładu, jeśli jest znany rozkład prawdopodobieństwa na zbiorze wartości danych wejściowych, czyli tzw. profil operacyjny)
N — liczba przebiegów programu

Wartość $\hat{R}$ szacuje się za pomocą testowania, ale związane jest to ze znacznymi trudnościami (wybór odpowiedniej strategii testowania, określonego zbioru danych testujących, długi czas testowania itp. [13]). W celu otrzymania wiarygodnego oszacowania $\hat{R}$ należy wykonać dużą ilość prób, w czasie których nie usuwa się wykrywanych błędów, co w rezultacie znacznie podnosi koszty tej metody. Mimo to, niektórzy badacze preferują modele w domenie danych ze względu na ich potencjalnie dużą dokładność i bezpośredni związek z operacją testowania programów [9, 13, 16]. Najczęściej jednak modele te stosuje się w powiązaniu z modelami pierwszej klasy, umożliwiającymi znacznie szerszą analizę niezawodnościową.

ROZSZERZONY MODEL NIEZAWODNOŚCIOWY PROGRAMU

W celu określenia modelu rozszerzonego przyjęto poniższe założenia.

Założenie 1. Współczynnik intensywności błędów programu jest wprost proporcjonalny do liczby błędów w programie

$$\lambda = C \cdot \varepsilon_r(\tau) \qquad (*$$

gdzie:

C — stała proporcjonalności [6]
$\varepsilon_r(\tau)$ — liczba błędów w programie po okresie testowania

Założenie 2. Po dostatecznie długim czasie testowania współczynnik przyjmuje wartość ustaloną $\lambda = \text{const}$.

Założenie 3. Błędy programu są wzajemnie niezależne i rozłożone równomiernie w całym programie.

Założenie 4. Wartości zmiennych wejściowych przy każdym przebiegu programu są wybierane losowo (aby uniknąć deformacji wyników przez wybór takich sekwencji wartości zmiennych wejściowych, z którymi jest związane wyższe albo niższe prawdopodobieństwo defektów).

Założenie 5. Wszystkie defekty programu są rejestrowane. Prawdopodobieństwo zarejestrowania defektu jest z reguły większe podczas fazy testowania niż podczas fazy użytkowania programu [8] (ze względu na specyficzne nastawienie personelu testującego), co prowadzi do bardziej pesymistycznej oceny niezawodności programu. Z praktycznego punktu widzenia jest to korzystne.

Założenie 6. Stopa błędów $\varrho(\tau)$ (liczba błędów na jednostkę czasu po okresie testowania τ [6]) maleje w sposób wykładniczy.

Przyjmijmy ponadto, że proces korekcji błędów programu może odbywać się błędnie, tzn. nie usuwa się wszystkich błędów powodujących dany efekt wykonania bądź wprowadza się nowe błędy. Powyższe założenie odróżnia niniejszy model od szeregu innych modeli niezawodnościowych.

W dalszej części artykułu przez t_o będziemy oznaczać czas operacyjny, czyli czas przebywania danego programu w systemie cyfrowym (mierzony czasem zajętości zasobów systemu, w szczególności zajętością jego jednostki centralnej). W ogólności jest on funkcją czasu kalendarzowego t i liczby programów w systemie.

Wykorzystując założenie 6 otrzymamy

$$\varrho(\tau) = \frac{d}{d\tau}\varepsilon(\tau) = a \cdot \varepsilon_r(\tau) = a[N - \varepsilon(\tau)]$$

Przy warunkach początkowych $\tau = 0$, $\varepsilon(0) = 0$ uzyskujemy

$$\varepsilon_r(\tau) = N[1 - \exp(-a \cdot \tau)]$$

Tak więc

$$\varepsilon_r(\tau) = N - \varepsilon(\tau) = N \exp(-a \cdot \tau)$$

Na podstawie podanych założeń przyjęto $a = \gamma - \beta$, czyli

$$\varepsilon_r(\tau) = Ne^{(\beta-\gamma)\tau} \qquad (**)$$

gdzie:

N — początkowa liczba błędów w programie
γ — współczynnik korekcji błędów programu
β — współczynnik nieskuteczności procesu korekcji

Na podstawie (*), (**) i ogólnej teorii niezawodności [3] uzyskamy

$$R(t_0) = \exp(-C \cdot \varepsilon_r(\tau)\, t_o) = \exp(-C \cdot N \cdot e^{(\beta-\gamma)\tau} \cdot t_0)$$

Zależność powyższa posłuży do wyznaczenia szeregu interesujących parametrów niezawodnościowych. W szczególności MTTF średni czas do wystąpienia błędu — ang. *Mean Time To Failure):*

$$MTTF = \int_0^\infty R(t_0)\, dt_0 = \frac{1}{C \cdot N} e^{(\gamma-\beta)\tau} = MTTF_0 \cdot e^{(\gamma-\beta)\tau}$$

Na rys. 1. widać wyraźny wzrost MTTF w funkcji czasu testowania. Rys. 2 przedstawia zależność $R(t_0)$ wykreśloną dla różnych czasów testowania τ.

Niezbędne wydłużenie czasu testowania programu $\Delta\tau$ w celu uzyskania wzrostu średniego czasu do wystąpienia błędu z $MTTF_1$ do $MTTF_2$ określimy na podstawie:

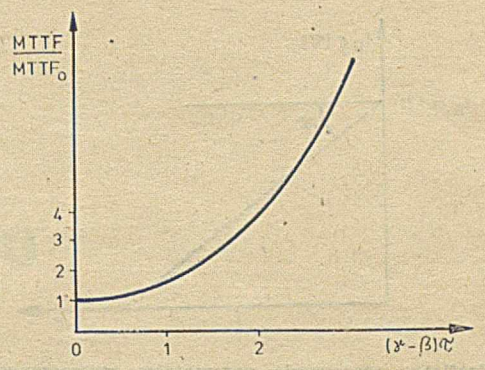
$$\Delta\tau = \frac{1}{\gamma - \beta} \ln \frac{MTTF_2}{MTTF_1}$$

Odpowiada to wykryciu Δn błędów programu

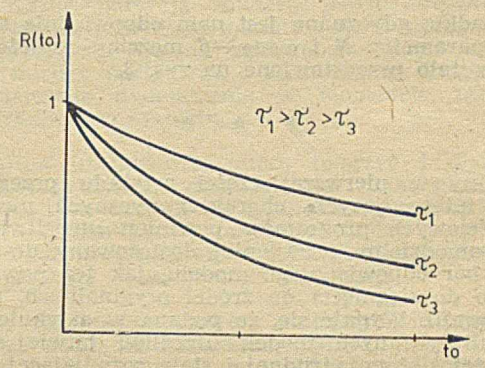
$$\Delta n = N_1 \left(1 - \frac{MTTF_1}{MTTF_2}\right)$$

gdzie:

N_1 — liczba błędów w programie po czasie testowania odpowiadającemu $MTTF_1$



Rys. 1. Zależność MTTF od czasu testowania



Rys. 2. Zależność niezawodności programu R (t_0) od czasu testowania

Interesujące może być przejście od czasu operacyjnego t_o do czasu kalendarzowego t.

Jeśli przez V^* oznaczymy współczynnik wykorzystania czasu maszyny

$$V^* = \frac{t_0}{t}$$

to niezawodność programu

$$R(t) = \exp\left(-\frac{C \cdot N}{V^*} e^{(\beta-\gamma)\tau} \cdot t_0\right) \qquad (***)$$

Ciekawy sposób powiązania czasu testowania z czasem kalendarzowym został przedstawiony w pracy [8]. Przyjmując, że

$$\beta - \gamma = a^* \cdot M$$

gdzie:

M — średnia liczba osób testujących i usuwających błędy w programie
a^* — stała proporcjonalności

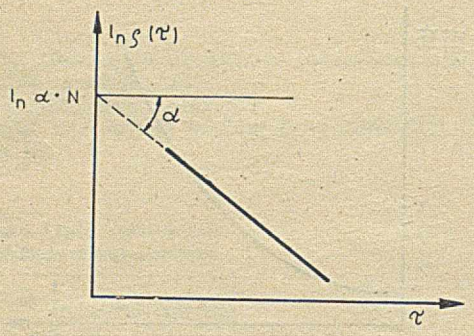
uzyskamy zależność między niezawodnością programu a liczebnością personelu testującego. W tym przypadku

$$R(t) = \exp\left(-\frac{C \cdot N}{V^*} e^{a^* \cdot M \cdot \tau} \cdot t_0\right)$$

W celu oszacowania niezawodności programu (***) konieczna jest znajomość poszczególnych parametrów modelu. Wartość stałej C określamy [6] z zależności:

$$C = \frac{\text{liczba błędów katastrofalnych w } (0, \tau)}{\text{liczba wykrytych błędów w } (0, \tau)}$$

Parametr V* wyliczymy wprost z definicji. Wyznaczanie parametrów N, β, γ, nie wymagające znajomości statystyki błędów, zostało przedstawione w [7]. Posłużono się przy tym pozytywną metodą próbkowania Jacksona.



Rys. 3. Graficzne szacowanie parametrów rozszerzonego modelu niezawodnościowego

W przypadku, gdy znana jest nam odpowiednia statystyka błędów, parametry N i $\alpha = \gamma - \beta$ możemy znaleźć graficznie, co zostało przedstawione na rys. 3.

* * *

Dokonany w pierwszej części artykułu przegląd najbardziej interesujących, charakterystycznych modeli niezawodnościowych programów powinien umożliwić czytelnikowi samodzielny i najlepiej dostosowany do jego potrzeb wybór odpowiedniego modelu, jak też powinien zachęcić go do sięgnięcia do źródeł oryginalnych, podanych w bibliografii. Wydaje się, że podana w artykule charakterystyka wybranych modeli umożliwi łatwiejszą orientację w tej złożonej, trudnej i stale rozwijającej się dziedzinie niezawodności oprogramowania.

Model zaproponowany w drugiej części artykułu ma — w porównaniu z już istniejącymi — pewne zalety, ułatwiające jego praktyczne wykorzystanie. Przede wszystkim umożliwia on rozpatrywanie procesu korekcji i testowania programu, w czasie którego nie usuwa się wszystkich błędów powodujących defekt wykonania bądź wprowadza się nowe błędy. Ponadto dopuszcza on możliwość usuwania za każdym razem więcej niż jednego błędu programu, co jest sytuacją najczęściej spotykaną w praktyce. Model umożliwia ponadto ocenę pracy zespołu testującego, poprzez określenie stosunku liczby błędów wprowadzonych dodatkowo w procesie korekcji do liczby błędów usuwanych, czyli stosunku β/γ. Poza tym wyznaczenie parametrów modelu może odbyć się bez znajomości dokładnej statystyki błędów, co czyni jego zastosowanie bardziej efektywnym, a ponadto umożliwia oszacowanie dokładności przewidywanych przezeń parametrów (dokładnie przedstawiono to w [7]. W pewnych przypadkach można wykorzystać też uproszczoną graficzną metodę wyznaczania parametrów, przedstawioną w artykule.

Podkreślić jednak należy, że zastosowanie tego modelu w praktyce wymaga jego potwierdzenia w szeregu realizacjach, co może znacznie przedłużyć okres jego wdrożenia.

LITERATURA

[1] Basin S. L.: Measuring the Error Content of Software. Science Applications, Inc., Paolo Alto, CA, September 1974, cyt. za [13]
[2] Ferdinand A. E.: A Theory of System Complexity. International Journal on General Systems, vol. 1, 1974
[3] Fidelis E., Firkowicz S.: Matematyczne podstawy oceny niezawodności. Wybrane zagadnienia, PWN, 1966
[4] Goel A. L.: Reliability and Other Performance Measures of Computer Software. II Proceedings of Relcomex'79, Zamek Książ, 1979
[5] Jeliński Z., Moranda P.: Software Reliability Research. w: Freiberger W.: Statistical Computer Performance Evaluation, Academic Press, New York, London, 1972
[6] Kornacki W.: Wybrane problemy analizy niezawodnościowej oprogramowania, INFORMATYKA, nr 11, 1980
[7] Kornacki W.: Szacowanie niezawodności maszyn cyfrowych. Raporty badawcze Instytutu Informatyki Politechniki Warszawskiej
[8] Musa J. D.: A Theory of Software Reliability and Its Application. IEEE Transactions on Software Engineering, nr 3, 1975
[9] Nelson E. C.: Software Reliability. TRW Software Series, TRW-SS-75-05, November 1975, cyt. za [13]
[10] Norma branżowa „Komputery. Niezawodność. Wymagania ogólne". BN-78, 3108-03
[11] Ottenstein L. M.: Quantitative Estimates of Debugging Requirements. Transactions on Software Engineering, September, 1979
[12] Schick G. J., Wolverton R. W.: Assessment of Software Reliability. Proceedings of Operation Research, Physica-Verlag, Wurzburg—Wienn, 1973
[13] Schick G. J., Wolverton R. W.: An Analysis of Competing Software Reliability Models. IEEE Transactions on Software Engineering, nr 2, 1979
[14] Schooman M. L.: Probabilistic Models for Software Reliability Prediction. w: Freiberger W.: Statistical Computer Performance Evaluation, Academic Press, New York, London, 1972
[15] Wagoner W. L.: The Final Report on a Software Reliability Measurement Study. Technology, Division, The Aerospace Corporation, El Sequado, CA, August 1973. cyt. za [13]
[16] Wilczyński W.: Struktura oprogramowania systemów komutacyjnych. Rozprawa doktorska, Politechnika Warszawska, 1980

Wszystkim Czytelnikom

najlepsze życzenia świąteczne i noworoczne

składa

Zespół Redakcji

STANISŁAW CHROBOT
Przedsiębiorstwo Systemów Komputerowych
MERA-SYSTEM
Warszawa

System operacyjny SOM-51 dla minikomputera MERA 400

SOM-51 jest pierwszą wersją handlową systemu operacyjnego SOM-5, opracowywanego dla minikomputera MERA 400 w Przedsiębiorstwie Systemów Komputerowych MERA-SYSTEM. Docelowo — różne wersje systemu operacyjnego SOM-5 tworzyć będą rodzinę systemów operacyjnych, przeznaczoną dla szerokiej gamy zastosowań.

Wersja SOM-51 wyposażona w interpreter języka MUMPS-400 pozwala wykorzystywać minikomputer MERA 400 do budowy systemów zarządzania z dyskową bazą danych dostępną w trybie konwersacyjnym z dużej liczby terminali. Baza danych zakładana jest w pamięci dyskowej MERA 9425, zaś jako terminale mogą być wykorzystywane drukarki znakowo-mozaikowe DZM-180/KSR, monitory znakowe MERA 7952 lub monitory grupowe MERA 7910. Pomocniczymi urządzeniami w systemie są czytanki i perforatory taśmy papierowej, drukarki znakowo-mozaikowe DZM-180 oraz pamięć na dyskach elastycznych.

W chwili obecnej SOM-51 eksploatowany jest wstępnie na trzech instalacjach komputerowych.

CHARAKTERYSTYKA JĘZYKA MUMPS

Język MUMPS jest językiem proceduralnym, porównywalnym z poziomem języka FORTRAN. W pierwszym rzędzie jest on zorientowany na przetwarzanie łańcuchów znakowych zmiennej długości. Dostępne są również standardowe operacje arytmetyczne i logiczne. Dane są reprezentowane w formie łańcuchów znakowych bądź w formie numerycznej. Język dopuszcza również operacje manipulowania bitami.

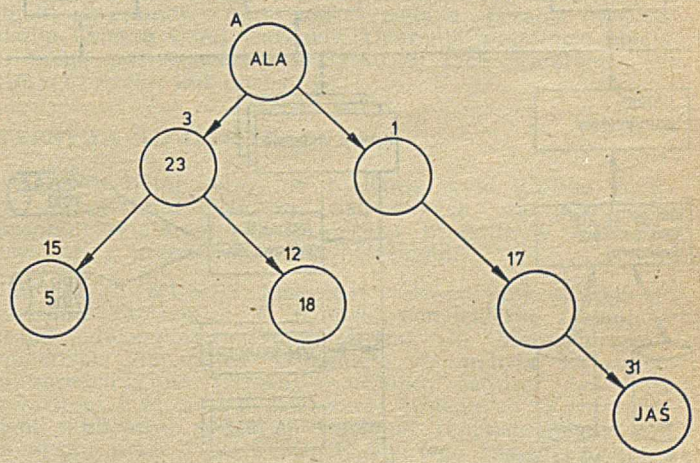
Wykonywanie programu MUMPS-owskiego jest czysto interpretacyjne. Każdy wiersz programu przetwarzany jest identycznie przy każdorazowym wykonaniu (kod pośredni nie jest tworzony). Interpreter języka działa w dwóch trybach: trybie wykonywania programu (typ pośredni) oraz w trybie redagowania programu (tryb bezpośredni). W trybie bezpośrednim programy mogą być redagowane, modyfikowane, testowane i przechowywane. Tryb pośredni pozwala na wykonywanie tych programów.

Na etapie generowania systemu pewna część pamięci operacyjnej przeznaczana jest na tzw. partycje użytkownika. Każda partycja zawiera jeden aktywny program użytkownika, jego dane lokalne i dane sterujące interpretacją programu. Nie ma żadnego ustalonego przyporządkowania między partycjami i terminalami. Przyporządkowanie następuje dynamicznie w momencie zgłaszania się użytkownika do systemu.

Program użytkownika może tworzyć i wykorzystywać dwa rodzaje danych: dane lokalne i dane globalne. Odwołania do tych danych są symboliczne, a pamięć przeznaczona dla nich jest w miarę potrzeby, przy czym dane lokalne rozmieszczane są w pamięci operacyjnej, natomiast dane globalne — w pamięci dyskowej.

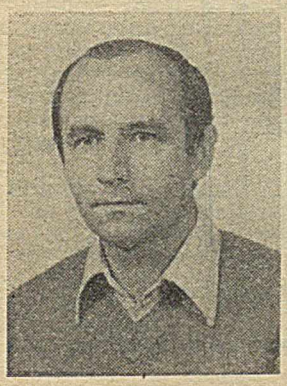
Dane lokalne są zbiorem zmiennych ustanowionych w obszarze działania konkretnej partycji i są dostępne wyłącznie w niej. Są to w zasadzie dane przejściowe lub robocze. Przyjmują one postać zmiennych prostych lub tablic jednowymiarowych.

Dane globalne, pamiętane na dysku w formie zbioru wielowymiarowych tablic, tworzą wspólną bazę danych dostępną dla wszystkich programów wewnątrz danej klasy użytkowników. Rekordy takiej bazy danych są traktowane jako elementy tablicy i są dostępne za pomocą indeksów.



SET ↑A = ALA
SET ↑A (3 15) = 5
SET ↑A (3,12) = 18
SET ↑A (3) = ↑A (3,15)+↑A (3,12)
SET ↑A(1,17,31) = JAŚ

Rys. 1. Przykładowa struktura tablicy globalnej A założona ciągiem instrukcji SET. Instrukcja SET przypisuje elementowi tablicy globalnej nową wartość. Jeśli element tablicy dotąd nie istniał, to zostaje on założony instrukcją SET



Dr inż. STANISŁAW CHROBOT ukończył w 1966 r. Wydział Łączności Politechniki Warszawskiej (obecnie Wydział Elektroniki) ze specjalnością Maszyny Matematyczne. Specjalizuje się w zakresie systemów operacyjnych. Jest współautorem systemów operacyjnych dla maszyn ZAM-41 (TRAN) i ODRA-1304 (WARS). Obecnie pracuje w Przedsiębiorstwie Systemów Minikomputerowych MERA-SYSTEM nad systemem operacyjnym SOM-5 dla minikomputera MERA 400.

Struktura tablic globalnych jest hierarchiczna, tzn. przyjmuje postać drzewa. Każdy element drzewa może mieć wartość numeryczną lub znakową i (lub) może być wskaźnikiem do niższego poziomu drzewa. Dane mogą być pamiętane na dowolnym poziomie. Nie istnieją żadne ograniczenia co do liczby wymiarów ani zakresu indeksów w obrębie dowolnego wymiaru. W dodatku liczba indeksów tablicy jest dynamiczna — tak, że jej zawartość i struktura mogą się zmieniać w czasie wykorzystywania (rys. 1). Obok zbiorów zmiennych globalnych dysk używany jest do przechowywania programów MUMPS-owskich i to zarówno usługowych programów systemowych jak i programów pisanych przez użytkownika. Dostęp użytkowników do programów i danych globalnych (globali) kontrolowany jest przez systemowy aparat protekcji.

W systemie może być zdefiniowanych kilka klas użytkowników. Każda klasą użytkowników ma dostęp tylko do tych programów i globali, które rezydują w tej klasie. Ponadto specjalnie oznaczone programy rezydujące w klasie nr 1 (tzw. klasie systemowej) dostępne są dla wszystkich użytkowników. W czasie zgłaszania się do systemu użytkownik podaje tzw. „kod klasy" (UCI), który otwiera mu dostęp do żądanej klasy w trybie pośrednim. Może on wtedy wywoływać programy, ale nie może ich zmieniać. Obok „kodu klasy" może być użyty dodatkowo „kod dostępu bezpośredniego" (PAC). Umożliwia on działanie w trybie bezpośrednim, w którym programy i globale określonej klasy mogą być tworzone i modyfikowane.

Do efektywnego wykorzystania systemu służy zbiór programów usługowych języka MUMPS. Spełniają one w systemie cztery funkcje:
— pomagają użytkownikowi utrzymać integralność bazy danych

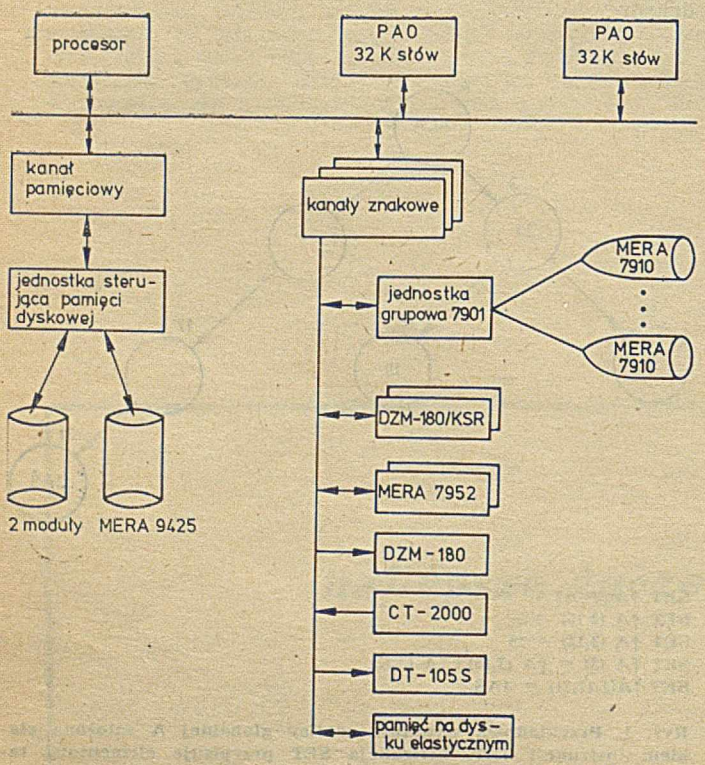
— umożliwiają kopiowanie zawartości dysku oraz przechowywanie i odtwarzanie poszczególnych programów i globali

— pozwalają zmieniać parametry systemu

— realizują standardowe operacje usługowe, takie jak drukowanie daty i czasu.

KONFIGURACJA SPRZĘTU

Podstawową konfiguracją umożliwiającą na obecnym etapie rozwoju sprzętu optymalne wykorzystanie systemu SOM-51 przedstawia rys. 2.



Rys. 2. Podstawowa konfiguracja systemu operacyjnego SOM-51

Jest to konfiguracja jednoprocesorowa z pamięcią operacyjną o pojemności 64 K słów, z dwoma modułami pamięci dyskowej MERA 9425 (10 KB) działającymi w kanale pamięciowym oraz z następującym zestawem urządzeń działających w kanałach znakowych:
— monitory ekranowe MERA 7910 przyłączone do jednostki grupowej 7901
— monitory ekranowe znakowe MERA 7952
— drukarki znakowo-mozaikowe z klawiaturą DZM-180/KSR
— drukarki znakowo-mozaikowe DZM-180
— czytnik taśmy papierowej CT-2000
— dziurkarka taśmy papierowej DT-105S
— dysk elastyczny.
W konfiguracji tej na partycje użytkowników wykorzystać można ok. 20 K słów pamięci operacyjnej, co pozwala na wygodne zorganizowanie kilkunastu partycji.

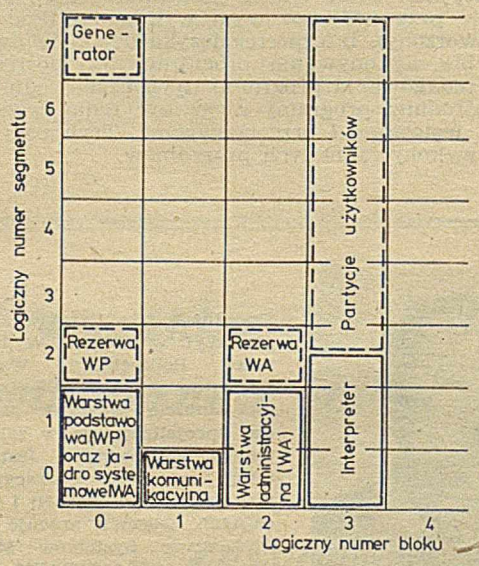
Wobec występujących trudności sprzętowych przygotowano mutację systemu, pozwalającą wykorzystywać konfigurację zmniejszoną, z pamięcią operacyjną 32 K słów. Mutacja ta nie obejmuje monitorów ekranowych MERA 7910, a na partycje użytkowników zostawia jedynie ok. 6 K słów pamięci operacyjnej. Biorąc pod uwagę mocno rozbudowany aparat nakładkowania programów MUMPS-owskich, można w ramach tej pamięci zorganizować dość wygodnie 6—8 partycji.

Równolegle trwają prace nad mutacją systemu dla konfiguracji zwiększonej. Dotyczą one wykorzystania pamięci operacyjnej większej niż 64 K słów, dwuprocesorowości i zabezpieczenia antyawaryjnego systemu, włączenia do systemu nowych urządzeń zewnętrznych, takich jak: pamięć na taśmach magnetycznych, multipleksor, drukarka wierszowa i urządzenia specjalizowane.

STRUKTURA OPROGRAMOWANIA

System Operacyjny SOM-5 jest systemem modularnym zarówno w sensie statycznym, jak i dynamicznym. Moduły statyczne, zwane segmentami programowymi są modułami okresu translacji i znikają w procesie konsolidacji. Moduły dynamiczne, zwane warstwami, są modułami okresu działania systemu (tzn. są rozróżnialne po wprowadzeniu do pamięci operacyjnej). Warstwa składa się z segmentów i z punktu widzenia procesu translacji i konsolidacji tworzy niezależny program. Układowy i programowy aparat dynamicznej protekcji warstw zapobiega rozprzestrzenianiu się błędów — z warstw uruchamianych lub „uszkodzonych" w wyniku awarii pamięci operacyjnej — do innych warstw. Aparat dynamicznej protekcji służy lepszej lokalizacji błędów pojawiających się w czasie pracy systemu, co z kolei jest podstawą do osiągnięcia zwiększonej niezawodności systemu.

W każdej warstwie może być uruchomiona pewna liczba procesów sekwencyjnych. Procesy sekwencyjne wszystkich



Rys. 3. Rozkład warstw SOM-51 w pamięci operacyjnej minikomputera MERA 400

warstw działają współbieżnie. Synchronizację procesów współbieżnych oraz dynamiczną protekcję warstw zapewnia jądro systemu operacyjnego IWA.

Mutacja systemu SOM-51 dla konfiguracji podstawowej składa się z czterech warstw:

- **warstwy podstawowej,** realizującej transmisje blokowe danych między pamięcią operacyjną i urządzeniami zewnętrznymi
- **warstwy komunikacyjnej,** realizującej protokół komunikacyjny dla monitorów ekranowych MERA 7910
- **warstwy administracyjnej,** organizującej zarządzenie dyskową bazą danych i standaryzującej operacje we/wy dla różnych urządzeń zewnętrznych
- **warstwy interpretera MUMPS,** wykonującego programy użytkowników umieszczone w partycjach.

Rozkład warstw tej mutacji w pamięci operacyjnej przedstawia rys. 3. Poniżej zostaną natomiast scharakteryzowane: jądro systemu operacyjnego oraz warstwy systemowe.

JĄDRO SYSTEMU OPERACYJNEGO IWA

W docelowym założeniu jądro systemu operacyjnego IWA ma być jądrem rodziny strukturalnych systemów operacyjnych dla dwuprocesorowego minikomputera MERA 400, działającego w trybie interakcyjnym w sieciach komputerowych o zwiększonej niezawodności. IWA jest zespołem programów działających na tzw. surowej maszynie. Przekształca ona jeden lub dwa procesory fizyczne w zbiór współbieżnie działających procesorów wirtualnych (symulowanych), z których każdy realizować może sekwencyjny proces programowany. Oczywiście, aby procesor wirtualny mógł działać, należy mu przydzielić czas procesora fizycznego (mówiąc inaczej — należy na niego przełączyć procesor fizyczny), ale w skali czasu większej od średniego czasu przełączenia możemy odnosić wrażenie, że wszystkie procesory wirtualne działają w sposób ciągły.

Maksymalna liczba procesorów wirtualnych wynosi 128. Programy procesorów współbieżnych muszą mieć hierarchiczną strukturę warstwową. Oznacza to, że całość oprogramowania powinna zostać zdekomponowana na moduły zwane warstwami (w skrajnym przypadku liczba warstw może oczywiście wynosić 1). W zbiorze tych warstw powinna zostać zdefiniowana hierarchiczna relacja dostępności. IWA daje możliwość definiowania maksymalnie 16 warstw rozłożonych na maksymalnie ośmiu poziomach hierarchii. Integralność warstw oraz relacja dostępności warstw kontrolowane są w czasie działania systemu (protekcja dynamiczna). Technicznie osiągane jest to w ten sposób, że każda wprowadzona do pamięci operacyjnej warstwa zajmuje własny logiczny blok pamięciowy. Zapobiega to niezamierzonej ingerencji jednej warstwy w program drugiej warstwy. Zamierzona współpraca warstw odbywa się za pośrednictwem jądra IWA.

Jeśli proces działający w warstwie *A* wymaga wykonania operacji *P* z warstwy *B* to wywołuje on ekstrakodową operację IWY — WOŁAJ WARSTWE z parametrami określającymi warstwę *B* i operację *P*. IWA sprawdza, czy warstwa *A* ma prawo wywołać warstwę *B* (czy *B* jest dostępne w *A*) i przy pozytywnym wyniku sprawdzenia przekazuje sterowanie do warstwy *B*. Operacja *P* kończy się ekstrakodową operacją IWY — WROC Z WARSTWY.

Z warstwami związano również problem regionów krytycznych (ang. **critical regions**), czyli wzajemnego wykluczania się w czasie operacji procesów współbieżnych działających na wspólnych zmiennych. IWA daje mianowicie możliwość definiowania dwóch rodzajów warstw: monitorowych i niemonitorowych.

Operacje warstwy niemonitorowej mogą być wykonywane jednocześnie lub mogą się przeplatać, zaś operacje warstwy monitorowej muszą się wykonywać rozłącznie w czasie. Technicznie jest to osiągane przez związanie z warstwą monitorową pojęcia regionu krytycznego i wstrzymywania przekazania sterowania do warstwy, jeśli jej region krytyczny jest zajęty.

Procesy współbieżne mogą się synchronizować wzajemnie za pomocą ekstrakodów OPOZNIJ i WZNOW realizowanych w IWIE a wywoływanych z warstw monitorowych. Synchronizacja procesów współbieżnych z działaniem urządzeń zewnętrznych odbywa się za pomocą ekstrakodu OPOZNIJ wywoływanego przez proces współbieżny i operacji WZNOW i WROC, wywoływanej na zakończenie obsługi przerwania od urządzenia zewnętrznego.

WARSTWA PODSTAWOWA

Warstwa podstawowa jest wyróżnioną, uprzywilejowaną warstwą systemu operacyjnego i wraz z jądrem systemowym rezyduje w zerowym (systemowym) bloku pamięci operacyjnej. Warstwa ta występować musi jako jedyna na najwyższym poziomie hierarchii w warstwowej strukturze systemu.

Zawiera ona oprogramowanie podstawowe wszystkich urządzeń wchodzących w skład konfiguracji systemu komputerowego. Pod pojęciem oprogramowania podstawowego urządzeń rozumie się następujący zakres funkcji:

- szeregowanie dostępu procesów współbieżnych do urządzenia
- inicjowanie transmisji danych między urządzeniem i pamięcią operacyjną
- obsługę przerwań zgłaszanych przez urządzenie
- rozpoznawanie sytuacji awaryjnych urządzenia.

Z powyższego wynika, że oprogramowanie podstawowe nie obejmuje:

— obsługi operatorskiej urządzeń
— zabezpieczenia antyawaryjnego urządzeń
— unifikacji strumieni informacyjnych różnych typów urządzeń.

Poza oprogramowaniem urządzeń fizycznych warstwa podstawowa zawiera programowy aparat służby czasu, rejestrujący czas bieżący i organizujący opóźnienie czasowe procesów współbieżnych oraz bufory komunikacyjne do wymiany danych między procesami współbieżnymi w obrębie pamięci operacyjnej.

Z zewnątrz warstwa podstawowa widziana jest jako zbiór definicji typów obiektów programowych. Obiektem programowym jest zbiór danych (zwanych reprezentacją obiektu) definiujący stan obiektu oraz zbiór operacji zwanych operacjami wejściowymi, za pomocą których procesy współbieżne wyższych warstw systemu oddziaływują na ten obiekt. Oddziaływanie to jest możliwe po wcześniejszym zainicjowaniu obiektu. Czynności tych dokonuje operacja inicjująca.

Typowym przykładem obiektu programowego, którego typ definiuje warstwa podstawowa jest obiekt związany z konkretnym urządzeniem zewnętrznym. Wykonanie operacji inicjującej tworzy w pamięci operacyjnej pewien zbiór danych, który opisuje stan urządzenia, zaś operacje wejściowe pozwalają procesom współbieżnym oddziaływać na to urządzenie, na przykład realizować transmisje między urządzeniem a pamięcią operacyjną. Obiekt programowy może być zainicjowany tylko jeden raz.

Po zainicjowaniu obiekt staje się dostępny w warstwie, która go zainicjowała. W warstwie, w której dostępny jest obiekt programowy może być wywołana operacja udostępniania obiektu innej warstwie. Operacje wejściowe obiektu mogą być wywoływane tylko w tych warstwach, w których obiekt jest dostępny.

WARSTWA ADMINISTRACYJNA

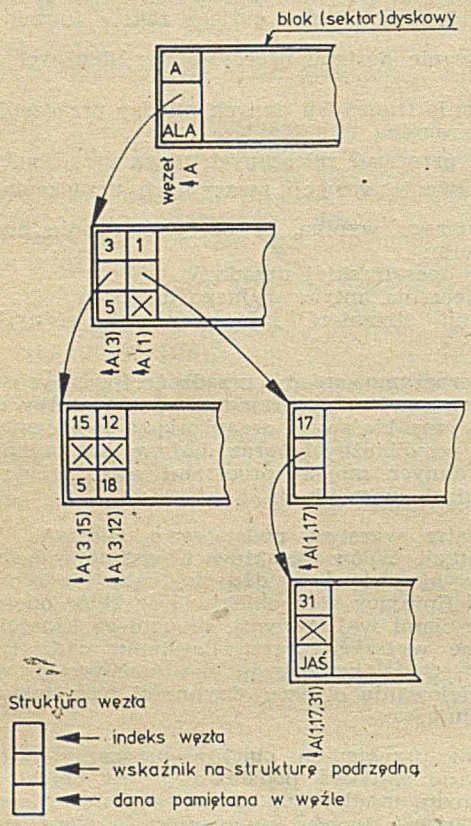
Warstwa administracyjna realizuje dwie główne funkcje: zarządzanie dyskową bazą danych oraz unifikację sposobu współpracy interpretera i programów MUMPS-owskich z urządzeniami zewnętrznymi.

W języku MUMPS klasyczny sekwencyjny dostęp do rekordów zapisanych w pamięci pomocniczej zastąpiony został odwołaniami symbolicznymi do hierarchicznych zmiennych globalnych. Osiągnięto to metodą logicznego odwzorowania na fizycznym nośniku pamięci dyskowej, zawartości i struktury drzewowej zmiennych. Każdy element tablicowej zmiennej globalnej reprezentowany jest na dysku przez tzw. węzeł. Węzły tego samego poziomu hierarchii umieszczane są kolejno (obok siebie) w bloku (sektorze) dyskowym lub w łańcuchu takich bloków.

Węzeł zawiera daną, informację o typie tej danej (dana znakowa lub numeryczna) i ewentualnie wskaźnik do bloku początkowego łańcucha zawierającego elementy tablicy niższego poziomu (rys. 4).

Bloki pamięci dyskowej na potrzeby bazy danych przydzielane są dynamicznie. Adresy wolnych bloków dyskowych są zapisane na tzw. mapach bitowych, w których każdemu blokowi odpowiada określony bit. W przypadku przydzielenia bloku początkowego lub kontynuacyjnego

wyszukiwany jest blok najbliższy bieżącemu. Zapewnia to minimalizację czasu wyszukiwania konkretnej danej. Przy aktualizacji danej dokonuje się przepakowania lub reorganizacji poszczególnych elementów wewnątrz łańcucha, aby zapewnić maksymalne wykorzystanie pamięci dla danych zmiennej długości.



Rys. 4. Odwzorowanie struktury tablicy globalnej A założonej na rys. 1 w pamięci dyskowej

W celu skrócenia czasu odwołań do zmiennych globalnych stworzono możliwość używania tzw. odwołania skróconego. Zawiera ono informacje o położeniu nowego węzła względem węzła wykorzystywanego ostatnio (a nie informacje o bezwzględnym położeniu węzła, jak to ma miejsce przy tzw. pełnym odwołaniu do zmiennej).

Bloki zawierające skasowane zmienne globalne lub ich części odzyskiwane są przez specjalny program tzw. Garbage Collector [1]. W czasie kasowania zmienne takie wstawiane są do łańcucha tego programu. Garbage Collector rozpakowuje taki łańcuch i uaktualnia mapy bitowe o adresy odzyskanych bloków.

Równolegle z zarządzaniem dyskową bazą danych warstwa administracyjna organizuje tzw. tablicę urządzeń dla interpretera. Tablica ta przyporządkowuje numery logiczne konkretnym urządzeniem zewnętrznym i ujednolica sposób wykorzystania tych urządzeń w interpreterze i programach MUMPS-owskich.

Dla wszystkich urządzeń zdefiniowana jest jedna postać rekordu danych, przesyłanego pomiędzy urządzeniem a programem, oraz zestaw instrukcji do zajmowania urządzenia, przesyłania danych, programów i znaków sterujących urządzeniem.

WARSTWA INTERPRETERA

Zasadnicze kryterium przy podziale funkcji między warstwą interpretera i warstwą administracyjną jest takie, że warstwa administracyjna zawiera aparat wspólny dla wielu (bądź wszystkich) partycji (np. tablica urządzeń we/wy, zarządzanie bazą danych) zaś warstwa interpretera, poza główną pętlą interpretacyjną — aparat prywatny partycji, np.: obsługa zmiennych lokalnych partycji, obsługa stosu sterującego partycji itp. W wyniku tego możliwe jest współdziałanie kilku warstw interpretera z jedną warstwą administracyjną, a co za tym idzie — wykorzystanie przez programy MUMPS-owskie pamięci operacyjnej większej niż 64 K słów.

Jak już wspomniano wyżej, wykonanie programu MUMPS-owskiego jest w pełni interpretacyjne i żaden kod pośredni nie jest generowany. Każde zdanie programu przetwarzane jest identycznie przy każdym wykonaniu. Dzięki temu prawie cały repertuar instrukcji MUMPS-owskich dostępny jest w obu trybach pracy interpretera (bezpośrednim i pośrednim).

PROGRAMY USŁUGOWE

Programy usługowe są to programy napisane w języku MUMPS i dostarczane jako część pakietu systemowego. Dzielą się one na dwie grupy: programy operatorskie i programy standardowe. Obie grupy rezydują na dysku i są zakatalogowane w klasie systemowej. Jednakże programy operatorskie mogą być wywołane wyłącznie z klasy systemowej, natomiast programy standardowe są programami ogólnodostępnymi, tzn. można je wywołać w każdej klasie użytkowników.

Programy operatorskie mają za zadanie pomóc operatorowi systemu w utrzymaniu bazy danych. Programy standardowe realizują funkcje najczęściej spotykane w programach zastosowaniowych lub funkcje związane z uruchamianiem programów i przechowywaniem bazy danych.

---

[1]) — Brak nazwy polskiej; jest to program scalający rozdrobnione fragmenty pamięci operacyjnej w większe całości

Kongres AFCET INFORMATIQUE 1980

Francuskie Stowarzyszenie Cybernetyki Ekonomicznej i Technicznej AFCET (Association Française pour la Cybernétique Économique et Technique) zorganizowało w Paryżu w dniach 24—27 listopada br. swój doroczny kongres AFCET INFORMATIQUE.

Jego przewodnim hasłem był przegląd aktualnych osiągnięć w zakresie sprzętu, oprogramowania, zastosowań oraz ich implikacji społeczno-gospodarczych. Organizatorzy skoncentrowali szczególną uwagę na dwóch nowych punktach programu, którymi były:

- jednodniowe seminaria ukierunkowane na najbardziej aktualne tematy (język ADA, specyfikacje systemowe, systemy rozłożone, kryptografia i ochrona danych, rozpoznawanie mowy)
- prezentacje audiowizualne dotyczące projektowania wspomaganego komputerem, grafiki komputerowej, nauczania oraz sztuki komputerowej (w tym koncert „Muzyka i obrazy cyfrowe).

Spis treści rocznika 1980

ARTYKUŁY PROBLEMOWE — nr — str.

BAŃKOWSKI JACEK, FABER ROMAN, FIAŁKOWSKI KONRAD, FURMAN MARIAN, KARPIŃSKI KONRAD, MURASZKIEWICZ MIECZYSŁAW, NOWICKI ZBIGNIEW, SOLAK JERZY — Uniwersalny pakiet przetwarzania i wyszukiwania informacji dla MERY 400 — 9 — 15

BEDNARZ JERZY, MAJEWSKI MAREK — Przygotowanie użytkownika bazy danych — 11 — 20

BELOWSKI JERZY, LIGUZ RYSZARD, STANASZEK TOMASZ — Oprogramowanie zautomatyzowanego systemu kierowania ruchem lotniczym GAMMA — 7—8 — 18

BOBROWSKI JAN (oprac.) — CURRICULUM '78 — program kształcenia informatyków w szkołach wyższych — 2 — 22

BRACHA JERZY, FORTECKI JERZY, KUBALE MAREK — ROZA — system układania harmonogramów zajęć — 10 — 14

BRADY PHILIP R., RUDAK BRONISŁAW, RUDAKOWA BARBARA — Połączenie lokalne dwu maszyn cyfrowych ODRA 1305 — 10 — 4

BUJKO JAN, STYCZYŃSKI ZBIGNIEW — Zasady archiwizacji w systemie projektowania wspomaganego komputerem — 11 — 13

BYLICKI ANDRZEJ — Szybkość efektywna transmisji synchronicznej — 10 — 24

CHROBOT STANISŁAW — System operacyjny SOM-51 dla minikomputera MERA 400 — 12 — 15

CICHOCKI WOJCIECH, NAHORSKI ZBIGNIEW, STEFAŃSKI JACEK, STUDZIŃSKI JAN — Postprocesor dla tokarek w systemie projektowania obróbki EXAPT — 7—8 — 22

CZERNIAK ZBIGNIEW, NIKODEMSKI MAREK — System operacyjny CROOK dla MERY 400 — 6 — 20

DĄBKOWSKI ANDRZEJ — Modelowanie systemowe uwarunkowań rozwoju społeczno-gospodarczego gmin — 4 — 13

DMOCHOWSKI WIESŁAW — Niezawodność eksploatacyjna systemów komputerowych ODRA 1305 i EC 1032 — 4 — 4

DOBIECKI AUGUSTYN — System informatyczny inżynierów ruchu — 5 — 10

DOBRUCKI LESZEK — Wybór systemu zarządzania bazą danych dla INFOCHEMU — 11 — 16

DRABENT WŁODZIMIERZ — Definiowanie języków programowania — Druga Metoda Wiedeńska

składnia — 10 — 7

semantyka — 11 — 7

DUBIELEWICZ ARTUR, RATAJCZAK JANUSZ — Monitor programowy dla systemu operacyjnego EGZEKUTOR E6RM — 5 — 13

DUDZIEWICZ MAGDALENA, POPIEL JAN — Demony w sieciowej bazie danych — 5 — 17

DZIENISZEWSKI JAN, RUSZKOWSKI JACEK — SYKLAM — system przetwarzania danych w klinice i poradni gastroenterologicznej — 7—8 — 30

ECKARDT THOMAS — Bank metod dla budowy prognoz z zastosowaniem minikomputera — 5 — 15

FRUNER STEFAN — Banki programów — nowa forma organizacji i eksploatacji oprogramowania użytkowego — 3 — 14

GACKOWSKI ZBIGNIEW — Informatyka w amerykańskiej dydaktyce — 9 — 12

GACKOWSKI ZBIGNIEW — Perspektywa komputeryzacji badania i projektowania systemów (język) — 12 — 4

GILECKI RYSZARD — Zarządzanie pamięcią z segmentacją — 3 — 17

GOGOŁEK SŁAWOMIR, GOGOŁEK WŁODZIMIERZ — Automatyzacja analizy i kodowania obrazów w daktyloskopii — 6 — 18

GOŁKA STANISŁAW — Systemy informatyczne w handlu i usługach motoryzacyjnych — 6 — 10

GOSPODAROWICZ ANDRZEJ — Możliwości automatyzacji układania harmonogramu zajęć — 10 — 12

GRENIEWSKI MAREK — Współczesne metody projektowania systemów informatycznych — 2 — 9

GWIAZDA JANUSZ — Perspektywy polskiej informatyki — 5 — 4

HAMMER CARL — Zarządzanie cyklem życia systemów — 7—8 — 4

JANICKI ANDRZEJ — Obiektowe systemy komputerowe — ważniejsze wyniki prac naukowo-badawczych — 3 — 4

JASTRZĘBSKI STANISŁAW — Metodyka projektowania programów w systemach konwersacyjnych — 5 — 7

KAMIŃSKI ALEKSANDER — Sprzęt RWPG wspomagający prace projektowe — 9 — 9

KISZKURNO ANDRZEJ, ŚWIECIŃSKI STEFAN — Doświadczalny system minikomputerowy na bazie elementów firmy INTEL — 6 — 13

KORNACKI WITOLD — Wybrane problemy analizy niezawodnościowej oprogramowania — 11 — 22

KORNACKI WITOLD — Wybrane modele niezawodnościowe programów — 12 — 11

KORPAL ANDRZEJ, KUBIAK ANDRZEJ — Architektura systemów wielomikroprocesorowych — 10 — 19

KOŚLACZ STANISŁAW, RZYMKOWSKI KRZYSZTOF — CAMAC — blokowy system elektroniczny do automatyzacji pomiarów i sterowania — 2 — 14

KRAMARCZUK JAN — Z myślą o informatyce lat osiemdziesiątych — 2 — 2

LANGER ADAM — Informatyka w gospodarce mieszkaniowej — 9 — 18

ŁAPIŃSKI ZDZISŁAW — Krajowy przemysł komputerowy — rzeczywistość i perspektywy — 1 — 4

MACHOWSKI ANTONI, STAROWICZ WIESŁAW — Doświadczenia z programowania w języku CSL ODRA 1300 — 2 — 17

MALLER V. A. J. — CAFS — pamięć masowa adresowana według zawartości — 9 — 4

MARCZYŃSKI ROMUALD — Mikroprocesory — rozwój i zastosowania — 3 — 11

MARKOWSKI TADEUSZ (oprac.) — Społeczeństwo informatyczne według V. Giscarda d'Estaing — 9 — 23

MURASZKIEWICZ MIECZYSŁAW, NOWICKI ZBIGNIEW — Sprzętowe przetwarzanie danych nienumerycznych — 4 — 20

MUSIAŁOWSKA ANNA, STABROWSKI MAREK — Oprogramowanie graficzne drukarki wierszowej — 6 — 15

NIEMYJSKA ELŻBIETA — Praktyczne aspekty stosowania metod strukturalnych — 2 — 11

PAWEŁCZYK ZBIGNIEW — Motywacyjne problemy wdrażania systemów informatycznych — 5 — 23

PAWELEC BRONISŁAW — Komputerowy system automatyzacji eksperymentów laboratoryjnych — 12 — 23

PIWOWAR BRONISŁAW — Rozwój sprzętu komputerowego MERY-ELWRO — 1 — 17

PRÓSZYŃSKI PIOTR — JAD-78 — język przetwarzania informacji sformatowanych — 9 — 20

	nr	str.
PUCHAŁKA TADEUSZ — Zadania i algorytmy komputerowego sterowania procesami technologicznymi	11	10
RAKOWSKI M. J. — Współpraca w ramach Jednolitego Systemu EMC. Rezultaty i perspektywy	3	20
ROCZNIAK JAN — Projektowanie programów strukturalnych metodą Jacksona	10	10
ROSZKOWSKI JERZY — Określenie i realizacja spójności językowej i informacyjnej systemów informatycznych	10	16
RUTKOWSKI STANISŁAW — Klimatyzacja ośrodków obliczeniowych	7—8	15
RYŻKO JAN, SIKORSKI ANDRZEJ — Holograficzne pamięci cyfrowe		
Część 1	6	4
Część 2	7—8	10
RZECZKOWSKI WIKTOR, SUBIETA KAZIMIERZ, — LINDA — system zarządzania bazą danych.		
Część 1	11	4
Część 2	12	8
SALAMON ZBIGNIEW J. — 20 lat MERY-ELWRO	1	13
SAWA RYSZARD — Automatyzacja procesów wydawniczych	7—8	26
SAWICKI WOJCIECH — Przydział procesora w systemach operacyjnych	4	17
SEDLÁČEK JIŘI — Oprogramowanie aplikacyjne czechosłowackich komputerów EC 1021 i EC 1025	4	23
SIKORSKI ANDRZEJ — Światłowody w systemach komputerowych	4	9
SOKOŁOWSKI ANDRZEJ — Przypadkowe zniszczenia lub przekłamania informacji	7—8	32
STEFANOWICZ BOGDAN — Metody heurystyczne w przetwarzaniu danych	2	20
SUKIENNIK JERZY, SZUMILEWICZ JERZY — R850 — nowy system firmy REDIFON	1	19
TABACZYŃSKI WŁODZIMIERZ — Analiza stanu zdrowia społeczeństwa	7—8	28
TEMLER ALEKSANDER — Informatyka w przemyśle motoryzacyjnym	6	3
TURSKI WŁADYSŁAW M. — Informatyka a społeczeństwo		
Część 1	1	3
Część 2	2	4
Część 3	3	7
ZIĘBIŃSKI MICHAŁ — FAKTURA — system kontroli fakturowania w handlu zagranicznym	5	20

USPRAWNIENIA, NOWE KONCEPCJE, POMYSŁY

Algorytm wykrywania i analizy zakleszczenia — **Jan Popiel**	1	27
System CAMAC w sterowaniu robotem przemysłowym — **Andrzej Gogolewski, Adam Żukowski**	2	27
INSEL — proste wybieranie informacji — **Henryk Bielinowicz**	3	36
Biblioteka programów standardowych dla urządzenia MERA 9150 — **Konrad Maruszczyk, Irena Zalewska**	6	36

PROBLEMATYKA BAZY DANYCH

Technologia rozproszonych baz danych. — oprac. **Marta Szymsiak**	1	38
Problemy budowy wspólnych baz danych — **Grzegorz Gruchman**	2	36
Regeneracja baz danych — **Leszek A. Maciaszek**	2	38
System zarządzania bazą danych i co dalej? — **Stanisław Mrozik**	3	39

NAUCZANIE I KSZTAŁCENIE

Współczesne tendencje szkolenia informatycznego — **Romuald Krzyżanowski**	3	32
O informatyzowaniu nauczania — **Mirosław Bereziński, Emanuel Czyżo, Wacław Zawadowski**	5	32
Informatyka w szkole dziesięcioletniej — **Mirosław Bereziński, Emanuel Czyżo, Wasław Zawadowski**	6	31
Metoda samokształcenia — **Władysław Klepacz**	6	33

Z KRAJU

Konferencja informatyków dojrzałych — **Krystyn Bernatowicz**	1	2
Uchwała IV Krajowej Konferencji Informatyków	1	3
INFOCHEM'79 — **Władysław Klepacz**	1	21
Strategia informatyzowania rachunkowości — **Adam B. Empacher**	1	22
III Branżowa Konferencja Ekonomistów Przemysłu Motoryzacyjnego — **Cz. Paczuła**	1	23
Komputeryzacja wielkich miast — WARSZAWA (oprac. **Z. G.**)	2	32
„Matematyka bliżej przemysłu, przemysł bliżej matematyki" **(A. B. E.)**	2	33
Komputerowa iluzja radomskiego „Waltera" — **(Z. G.)**	3	23
DOLMED — nowe oblicze diagnostyki **(K. B.)**	3	24
MERA-400 bez obsługi **(Z. G.)**	4	25
Ustronie informatyczne **(A. B. E.)**	4	30
INFOSEM'79 **(A. B. E.)**	4	31
Tak dużo zrobiono, tak wiele pozostało... — **Krystyn Bernatowicz**	5	26
Banki danych w centralnych systemach informatycznych (seminarium SPIS'79) — **Henryk Dąbrowski**	5	28
I Konferencja CAMAC'80 — **Janusz Zalewski**	6	22
Słudzy Królowej Nauk **(A. B. E.)**	6	25
Wymiana doświadczeń śląskich informatyków **(E. K.)**	6	26
Konferencja Informatyków POLMO — **Władysław Klepacz**	6	27
Łódzkie Stowarzyszenie Informatyków — **Z. Gluza**	7—8	34
CYBERIADA 2000 — **A. B. Empacher**	7—8	35
VIDEOTON polskiemu partnerowi **(Z. G.)**	7—8	36
Sprawa informatyki — **Zbigniew Gluza**	9	25
Filmowe konfrontacje — **Barbara Łukasik-Makowska**	10	27
Metody układania harmonogramu zajęć **(M. K.)**	10	28
Biblioteka oprogramowania użytkowego — **Elżbieta Kierczuk**	11	25
INTERBIURO'80 **(Z. G.)**	12	27
Informatyka w badaniach naukowych chemii — **Janusz Zalewski**	12	23

ZJEDNOCZENIE INFORMATYKI

Prace badawczo-rozwojowe w Zjednoczeniu Informatyki — **Grzegorz Głownia**	1	24
Koszalińskie Dni skutecznie propagują informatykę **(K. B.)**	1	26
XV lat ZETO Wrocław **(W. K.)**	2	25
Uroczystość jubileuszowa **(K. B.)**	2	26

	nr	str.
Krakowskie echa europejskich „Dieboldów" (K. B.)	3	26
System wyróżnień — Irena Malerczyk-Dańda, Zdzisław Głuszak	3	27
ZETO wobec wymagań bezpośredniego użytkownika — Janusz Gwiazda	4	26
Obniżka cen usług informatycznych — Jerzy Rej	4	28
Narada aktywu polityczno-gospodarczego (W. K.)	5	30
Budowa i organizacja eksploatacji systemów teleprzetwarzania — Tomasz Pawlak	5	31
Jeszcze raz o cenniku usług informatycznych — Jerzy Rej	5	31
Doświadczenia z wdrożenia kompleksowego systemu informatycznego — Mieczysław Ciszek, Stanisław Czyżowski	6	28
System planowania i rozliczania produkcji w Zakładach Przemysłu Odzieżowego DANA — Antoni Kowalczyk	7—8	37
Seminarium promocyjne Zjednoczenia Informatyki w ZSRR — Małgorzata Galoch, Marek Wejtko	7—8	38
Rozwój infrastruktury Zjednoczenia Informatyki (Z)	7—8	39
ZETO Jelenia Góra — nie tylko komputery — Krystyn Bernatowicz	7—8	40
Sztandar dla Szczecina, uznanie dla pozostałych (K. B.)	7—8	41
Kronika przeobrażeń — bilans dokonań ZETO Katowice — Krystyn Bernatowicz	9	29
Informatyka w Centralnej Składnicy Zaopatrzenia PKS — Stanisław Dąbkowski, Izabella Kowalik	9	32
Systemy informatyczne w Kombinacie Górniczo-Hutniczym Miedzi — Tomasz Tymeński, Waldemar Zagrajek	10	31
SETR — system dla przedsiębiorstw transportu samochodowego — Danuta Milewska, Barbara Wasil	10	32
EWGRUN — podsystem ewidencji gruntów — Michał Kosiński, Stanisław Zaremba	11	26
Egzekutory z uprawnieniami wyprowadzenia wydruków — Leonard Dobiecki	12	25
Po raz ostatni w Jeleniej Górze? (K. B.)	12	27
Zlaz Informatyków (I. K.)	12	27
CENTRUM ETOB		
ETOB — Katowice: 15 lat i Dzień Budowlanych (wład)	1	30
5-lecie naczelnego	1	31
10 lat ETOBU w Bydgoszczy (wład)	1	31
Odrowski remanent — Wincenty Łada	2	29
Kontrakty z ZETO	3	30
O cenniku usług informatycznych — Wincenty Łada	3	30
Na przełomie dekad — Wincenty Łada	4	32
Postulaty 1980 — Wincenty Łada	10	30
PORTRETY ZAWODOWE		
Leon Pesta (wład)	1	32
Gdzie diabeł nie może... (wład)	2	30
Bronisława Kościelska (wład)	3	31
Kazimierz Sowa (wład)	4	34
Androidy 1980 (wład)	5	25
ZWIĄZKI ZAWODOWE		
Uchwała I Walnego Zjazdu NSZZ Pracowników Informatyki	11	29
Niezależni i samorządni — Włodzimierz Grudziński, Ryszard Kasprzyk	12	31
Związek czy Stowarzyszenie?	12	32

	nr	str.
ZE ŚWIATA		
Rynek małych przedsiębiorstw zmienia oblicze przemysłu komputerowego — oprac. W. Klepacz	1	33
Ekspansja Japończyków — W. Tryuk	1	34
Międzynarodowe sympozjum Zastosowań Mikroprocesorów i Mikrokomputerów — A. Grandys, L. Namysłowski	1	35
System fonicznej informacji bankowej — S. Jasiorowski	1	35
Informatyka w RFN — Adam Rzymowski	1	36
Międzynarodowe konferencje (W. K.)	2	35
Konferencja COMPCONTROL'79 — Witold C. Dorosiński	3	28
Problemy rozwoju systemu ASGS (T. J.)	3	29
Małe systemy — nowa koncepcja ICL (T. J.)	3	29
Polityka firmy IBM w latach 1978—1979 — Stanisław Jasiorowski	4	35
Coraz sprawniejsze, coraz tańsze — Jarosław Deminet	5	34
Informacje różne — T. Jaegerman	5	36
CZŁOWIEK I KOMPUTER — konflikt czy współpraca? (W. K.)	6	34
Informacje różne (T. J.)	6	34
VI Międzynarodowy Kongres Informatyki Europejskiej w Wiedniu — Władysław Klepacz	7—8	42
Przemówienie Ministra Nauki i Badań Austrii, dr Herthy Firnberg	7—8	44
Ośrodek Obliczeniowy'80 — W. Łada	7—8	47
Najnowsze tendencje w dziedzinie komputerowych systemów sterowania — Janusz Zalewski	9	34
Stowarzyszenie austriackich informatyków (W. K.)	9	36
Międzynarodowa Konferencja nt. zastosowań ETO w budownictwie	10	34
Automatyczny skład wzorów matematycznych (W. K.)	10	34
Pismo arabskie w automatycznym składzie (W. K.)	10	35
Postulaty kształcenia informatyków w Austrii (W. K.)	11	31
Analizator mowy bez „przyuczenia" (A. R.)	11	33
Wrażliwość skomputeryzowanego społeczeństwa na zagrożenia (A. M.)	11	33
Komputeryzacja szklarni (W. K.)	11	34
Nowoczesne metody programowania dla niewidomych (W. K.)	11	35
Komputer w kościele (W. K.)	11	35
Szanse i granice przetwarzania informacji — Józef Piasecki	12	33
RECENZJE		
Jak zwiększyć wydajność systemów komputerowych? — Zbigniew Kisieliński	1	37
W poszukiwaniu rozwiązań optymalnych — Mieczysław Muraszkiewicz	2	34
O systemach minikomputerowych — Janusz Zalewski	3	34
Wokół „Metodologii programowania" — Andrzej Blikle	4	36
Słownik niemiecko-polski (A. B. E.)	5	38
Słownik niemiecko-polski (A. B. E.)	5	38
„Tysiąc" informatycznych słów — Włodzimierz Gogołek	5	40
„Lekarstwo" Bürgera — Adam B. Empacher	6	39

	nr	str.
Programowanie rozumne — **Antoni Mazurkiewicz**	7—8	49
Panorama informatyki — **Stanisława Bonkowicz-Sittauer**	9	37
O zastosowaniu minikomputerów w badaniach eksperymentalnych — **Janusz Zalewski**	10	36
Mini... mikro... — **Adam B. Empacher**	11	37
Kontrola w nowoczesnych systemach finansowo-księgowych — **Maria Waśko**	12	36

LISTY (TRYBUNA CZYTELNICZA)

Pod jakim szyldem pracuje informatyk? — **Elżbieta Kierczuk**	1	29
„Projektowanie" algorytmów" — **Bogdan Stefanowicz**	1	29
(Hiper)wtajemniczenie teoriografowe — **Maciej M. Sysło**	2	31
O niezawodności JS EMC — **Bolesław Gliksman**	4	39
O niezawodności sprzętu krajowego — **Wit Drewniak**	4	39
A wdrażać trzeba — **Elżbieta Kierczuk**	5	40
Czy uda się ożywić wymianę myśli na łamach INFORMATYKI? — **Krystyna i Jacek Żebrowcy**	11	36

TERMINOLOGIA

Przeciw zamętowi w polskiej terminologii — **Janusz Zalewski**	3	35
O jednolitą terminologię: „Przerwanie" — **Janusz Zalewski**	4	38
„Przerwanie" — **Janusz Zalewski**	5	37
„Bit" — **Janusz Zalewski**	6	38
„Pamięć" — **Janusz Zalewski**	7—8	50
„Sprzęg" — **Janusz Zalewski**	9	39
„Czas rzeczywisty" — **Janusz Zalewski**	10	37
„Instrukcja" czy „rozkaz"? — **Janusz Zalewski**	11	38
„Programowanie" — **Janusz Zalewski**	12	36

POGLĄDY

Doktoraty czy stopnie zawodowe — **Władysław M. Turski**	7—8	III okł.
Czy istnieje „polska szkoła informatyki"? — **Juliusz Lech Kulikowski**	9	III okł
O planowaniu w informatyce — **Janusz Gwiazda, Janusz Kwiek**	10	III okł.
Rachunkowość nie doinformatyzowana — **Tadeusz Peche**	11	40
Gdzie jest informatyka? — oprac. **Z. Gluza**	12	39

WARUNKI PRENUMERATY

Prenumeratę przyjmują oddziały RSW „Prasa-Książka-Ruch" i urzędy pocztowe.

Jednostki gospodarki uspołecznionej, instytucje, organizacje i wszelkiego rodzaju zakłady pracy zamawiają prenumeratę w miejscowych oddziałach RSW „Prasa-Książka-Ruch", a w miejscowościach, w których nie ma oddziałów — w urzędach pocztowych. Czytelnicy indywidualni opłacają prenumeratę wyłącznie w urzędach pocztowych i u doręczycieli.

Cena prenumeraty krajowej wynosi:

- kwartalna — 90 zł
- półroczna — 180 zł
- roczna — 360 zł

Przedpłaty przyjmowane są w następujących terminach:

- do 10 czerwca — na III kwartał i II półrocze
- do 10 września na IV kwartał
- do 25 listopada — na rok następny, I kwartał, I półrocze
- **do 10 marca — na II kwartał**

Prenumeratę ze zleceniem wysyłki za granicę przyjmuje Centrala Kolportażu Prasy i Wydawnictw, ul. Towarowa 28, 00-958 Warszawa, konto NBP oddział w Warszawie nr 1531-201045-139-11 w terminach obowiązujących dla prenumeraty krajowej. Prenumerata ta jest droższa od prenumeraty krajowej o 50% dla zlecenioodbiorców indywidualnych i o 100% dla zlecających instytucji i zakładów pracy.

Egzemplarze archiwalne czasopism wydawanych przez NOT -SIGMA można nabyć w Dziale Handlowym przy ul. Mazowieckiej 12, 00-048 Warszawa, tel. 26-80-16.

BRONISŁAW PAWELEC
Centrum Komputerowych Systemów Automatyki i Pomiarów
MERA-ELWRO
Wrocław

Komputerowy system automatyzacji eksperymentów laboratoryjnych

Współczesne laboratorium naukowo-badawcze charakteryzuje kompleks prowadzonych doświadczeń zarówno o charakterze podstawowym, jak i symulacyjnym. Symulacja rozumiana jako odtworzenie własności rzeczywistego procesu lub zjawiska za pomocą jego modelu przy użyciu sprzętu komputerowego, wraz z jego rozwojem, jest jedną z coraz szerzej stosowanych metod w badaniach naukowych, jak również w projektowaniu w różnych dziedzinach nauki i techniki. Procesy, które można przenieść wprost do laboratorium (dla przykładu badania wytrzymałościowe), służą jako modele do badań inżynierskich. Efekty tych zadań przenoszone są do projektowania.

Automatyzacja kompleksowa skomplikowanych procesów technologicznych z użyciem maszyn cyfrowych wymaga, na etapie analizy procesu i projektowania systemu, rozwiązania szeregu zagadnień, do których zastosowanie szczególnie metod i technik symulacji jest niezbędne. W tym względzie formy symulacji można zasadniczo podzielić na:

- ukierunkowane problemowo, wykorzystujące symulację jako metodę do rozwiązania problemu
- ukierunkowane procesowo, wykorzystujące maszynę cyfrową jako symulator.

Druga z form wymaga zaprojektowania algorytmu symulacji oraz jego przekształcenia w program odpowiedni do wymagań maszyny cyfrowej. Obydwie formy symulacji znajdują zastosowanie na etapie analizy procesu przy badaniu zależności dynamicznych w oparciu o modele matematyczne. Efekty symulacji wykorzystywane są w fazie projektowania systemu rzeczywistego.

Do realizacji zadań laboratoryjnych system symulacji oprócz tradycyjnego zestawu komputerowego (jednostka centralna z urządzeniami we/wy i z pamięcią) powinien posiadać:

- urządzenia sprzęgające ze stanowiskami laboratoryjnymi i z układami sterowania
- symulator procesu, dający możliwość wprowadzania sygnałów wejściowych, o odpowiedniej do danego procesu konfiguracji i strukturze jakościowej, oraz wyprowadzania sygnałów wyjściowych
- odpowiednio elastyczny system oprogramowania podstawowego i użytkowego.

Przedstawiona w dalszej części pracy koncepcja laboratoryjnego systemu wzorowana jest na opracowanych i wdrożonych systemach:

- symulacji prac laboratoryjnych w ROBRAPCh „Chemoautomatyka", Oddział Łódzki
- sterowania eksperymentem w Laboratorium Instytutu Materiałoznawstwa i Mechaniki Technicznej Politechniki Wrocławskiej
- automatyzacji prac laboratoryjnych w Laboratorium Instytutu Podstawowych Problemów Techniki PAN w Warszawie
- automatyzację prac laboratoryjnych dla Politechniki im. Kim Czaka w Phenianie (KRL-D).

Powyższe systemy opracowano w Pracowni Projektowania Systemów Centrum MERA-ELWRO przy udziale specjalistów zainteresowanych instytucji.

FUNKCJE I KONFIGURACJA SYSTEMU

Wstępne planowanie przebiegu eksperymentu w laboratorium odbywa się poza systemem. Jego wynik wprowadzany jest do systemu i przechowywany w pamięci operacyjnej. Stanowiska laboratoryjne wyposażone są zarówno w tradycyjną aparaturę typu mostek tensometryczny, woltomierz cyfrowy, czy falomierz, jak i w specjalizowane pulpity pomiarowe oraz monitory ekranowe, które służą do komunikacji z systemem. System automatyzacji prac laboratoryjnych zapewnia:

- bieżącą rejestrację analogowych i cyfrowych sygnałów pochodzących ze stanowisk pomiarowych
- przekształcenie odebranych sygnałów w jednostki fizyczne, odpowiadające parametrom kontrolowanych przebiegów
- obróbkę wyników pomiarów na bieżąco oraz w systemie off-line

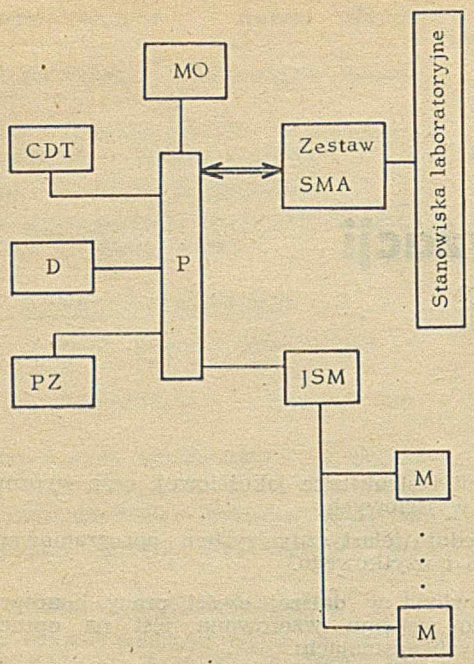
Pomiary przechowywane są w pamięci operacyjnej, bądź przenoszone do zbiorów w pamięci zewnętrznej typu taśmy, dyski magnetyczne

- sterowanie układów wykonawczych sygnałami wypracowanymi programowo i wyprowadzanymi poprzez wyjścia cyfrowe i analogowe
- dialog operatora z systemem w zakresie sterowania i sygnalizacji kolejnych etapów (kroków) eksperymentu
- automatyczne lub wymuszone (programowo) wyświetlanie wartości parametrów eksperymentu.

Poza powyższymi podstawowymi funkcjami w systemie mogą być realizowane funkcje ukierunkowane na określone zastosowania. Dla przykładu, w systemie zainstalowanym w KRL-D opracowany jest symulator analogowych obwodów regulacji przeznaczony przede wszystkim do celów dydaktycznych.

Urządzenia wchodzące w skład typowego systemu automatyzacji prac laboratoryjnych pokazano na rys. 1.

Dr inż. BRONISŁAW PAWELEC ukończył w 1967 r. Wydział Elektroniki Politechniki Wrocławskiej a w 1973 r. Zaoczny Wydział Matematyki Uniwersytetu Wrocławskiego. W 1980 r. doktoryzował się w Instytucie Cybernetyki Technicznej Politechniki Wrocławskiej. Pracuje obecnie w Pracowni Projektowania Systemów Centrum „MERA-ELWRO" we Wrocławiu na stanowisku starszego projektanta systemów cyfrowych. Zajmuje się komputerowymi systemami sterowania eksperymentów laboratoryjnych.



Rys. 1. Konfiguracja systemu laboratoryjnego
P — jednostka centralna ODRA 1325 zawierająca: pamięć operacyjną 32 k słów, kanał przemysłowy do obsługi sygnałów cyfrowych i analogowych, kanały znakowe do obsługi urządzeń typu czytnik taśmy i drukarka, kanały buforowane do współpracy z pamięcią taśmową i dyskową
MO — konsola operatora MO-325 przeznaczona do współpracy operatora z systemem
CDT — czytnik (dziurkarka taśmy papierowej CDT-325
D — drukarka znakowo-mozaikowa DZM 180/325
PZ — pamięć taśmowa PT3 lub dyskowa PDS325
JSM — jednostka sterowania monitorów ekranowych JSG7801
M — monitor ekranowy VT340-0, przeznaczony do wyświetlania wartości parametrów na stanowiskach pomiarowych
Zestaw SMA — zespół opracowanych w MERA-ELWRO urządzeń przeznaczonych do bezpośredniego zbierania sygnałów cyfrowych i analogowych ze stanowisk pomiarowych. Typowy zestaw SMA składa się z: bloku sterowania kanału przemysłowego, który organizuje transmisję z (do) procesora, oraz bloków funkcjonalnych wejścia/wyjścia cyfrowego i analogowego, zegara cyfrowego itd.

OPROGRAMOWANIE SYSTEMU

Oprogramowanie systemów laboratoryjnych zostało zaprojektowane w oparciu o metodologię MASE (Master Control Executive) opracowaną przez Vanghan Programming Servicing [1]. Praca systemu odbywa się pod kontrolą egzekutora EX2P, który narzuca określoną strukturę oprogramowania. Przykładową strukturę pokazano na rys. 2.

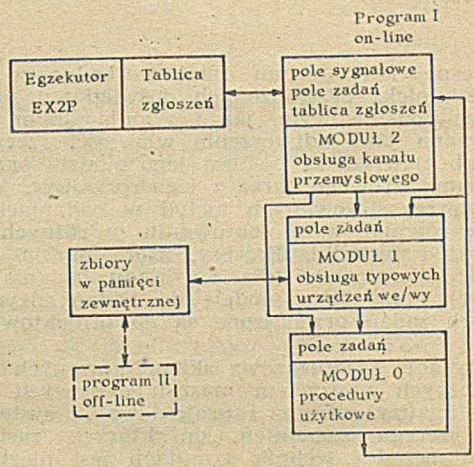
Program współpracujący z kanałem przemysłowym składa się z dwóch lub trzech modułów, z których jeden jest priorytetowym (moduł 2). Przez moduł rozumie się tu odrębną funkcjonalnie część programu o następujących własnościach:
— powołanie modułu do pracy następuje z innego modułu lub na sygnał zewnętrznego przerwania
— zawieszenie pracy modułu odbywa się na rozkaz wewnętrzny
— przy powołaniu do pracy kilku modułów działa ten, który ma najwyższy priorytet.

Moduł 2 (priorytetowy) programu jest modułem wyróżnionym, posiadającym stopień zaufania klasy T. Realizuje on obustronny kontakt z blokami SMA poprzez programowanie akcji egzekutora. Wyniki transmisji z (do) kanału przemysłowego ustawiane są w polu sygnałowym, skąd odbierane są przez moduł 2. Po zidentyfikowaniu odebranego sygnału podejmowana jest odpowiednia akcja związana z ustawianiem określonego zadania do modułów 1 i 0.

Moduł 1 programu służy do komunikacji z urządzeniami zewnętrznymi typu: czytnik/dziurkarka taśmy papierowej, dziurkarka znakowo-mozaikowa, monitory ekranowe, pamięci zewnętrzne, itd. Zadania dla tego modułu zlecają moduły 2 i 0. Do usprawnienia pracy modułu założono prowadzenie równoległej transmisji. Konsekwencją tego jest ustawianie kolejek inicjacji transmisji do urządzeń zewnętrznych, co wprowadza priorytety urządzeń i zwiększa szybkość obsługi.

Moduł 0 jest modułem użytkowym, do którego dołączane są procedury realizujące użytkowe funkcje systemu. Procedury te wprowadzane są wprost do pamięci operacyjnej albo do pamięci zewnętrznej, w przypadku, gdy zakłada się pracę nakładkową. Nie ma tu ograniczeń co do języka programowania, w którym procedury są przygotowane.

Komunikacja między modułami programu odbywa się poprzez zadania (są to wewnętrzne rozkazy programu), ustawiane w odpowiednich polach zadań.



Rys. 2. Ogólna struktura systemu programowego

Na rys. 2 zaznaczono program II, który znajduje się w pamięci operacyjnej i uruchamiany jest okresowo na specjalne żądanie. Funkcje tego programu ograniczone są do sprawozdawczości.

* * *

Proponowany tu system został poprzedzony szczegółową analizą potrzeb laboratorium naukowo-badawczego oraz możliwości sprzętu i oprogramowania podstawowego i użytkowego, przy czym uwzględniono rozwiązania już istniejące [2, 3, 4]. Zastosowanie tego typu systemów wydaje się być szczególnie cenne tam, gdzie chodzi o umożliwienie rejestracji i kontroli parametrów szybkozmiennych procesów, a tradycyjne metody zawodzą. Przykładem może być system prezentowany w [3], w którym zastosowano dwa specjalne bloki wejść analogowych nie zawierające komutatorów stykowych. Bloki te umożliwiają odczyt wartości pomiarowych z częstotliwością około 1500 Hz, co pozwala rejestrować szybkozmienne przebiegi i przenosić je do zbiorów na taśmach lub dyskach magnetycznych z przeznaczeniem do dalszej obróbki.

W zakończeniu należy podkreślić, że prezentowany system można zrealizować również na innym sprzęcie (np. MERA 400), pod warunkiem, że sprzęt ten ma urządzenia sprzężenia ze stanowiskami laboratoryjnymi.

LITERATURA

[1] A significant Advance in Programm Technology for on-line Industrial Control Systems. Automation. September 1971
[2] Kordecki H., Pawelec B., Więckiewicz J.: Analiza wykorzystania systemu komputerowego ODRA 1325/SMA w warunkach przemysłowych i laboratoryjnych. Prace VII KKA Rzeszów, 1977
[3] Kordecki H., Pawelec B., Więckiewicz J.: Prowadzenie eksperymentów z wykorzystaniem systemu ODRA-SMA w warunkach laboratoryjnych. Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej nr 41, 1976
[4] Paweleć B. i inni: Symulacja procesów chemicznych w oparciu o system ODRA-SMA. Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej nr 41, 1976
[5] Schone A.: Simulation technischer Systeme. Karl Hanser Verlag. 1974.

Egzekutory z usprawnieniami wyprowadzania wydruków

Znakomita część użytkowników komputerów serii ODRA 1300 eksploatuje swoje systemy pod kontrolą egzekutora. Dlatego też wszelkie usprawnienia tego podstawowego systemu operacyjnego są przez nich na ogół mile widziane.

Praktyka wykazuje, że praca wieloprogramowa, zagwarantowana przez standardowy egzekutor, bywa często poważnie ograniczona. Główną przyczyną tego stanu rzeczy jest blokowanie wolnych urządzeń zewnętrznych przez programy wykorzystujące je w sposób nieefektywny. Przykładem może być grupa programów kontrolnych, które wyprowadzają z reguły niewielkie tabulogramy na drukarkę wierszową, uniemożliwiając wykorzystanie jej przez inne programy użytkowe. Sprawa komplikuje się jeszcze bardziej w wypadku awarii tychże urządzeń; wówczas praca większości programów korzystających z wolnych urządzeń peryferyjnych staje się całkowicie niemożliwa.

Problem ten może być rozwiązany przez prosty system wspomagający egzekutor, który umożliwia uzależnienie programów od innych, znacznie szybszych urządzeń. Szczególnie duże efekty daje zastąpienie drukarki wierszowej jednostką magnetycznej pamięci taśmowej. Jeśli weźmie się pod uwagę, że wypisywanie wydruków z taśmy może być dokonane na komputerze innego typu, wówczas efekty mogą być jeszcze większe.

Źródłem zahamowań pracy komputera jest także komunikacja między nim i operatorem, odbywająca się za pośrednictwem monitora. Spowodowane to jest m. in. skomplikowaną obsługą pewnych czynności operatorskich, wymagających częstego powtarzania, oraz blokowaniem monitora przez komunikaty egzekutora. W tym przypadku dokonanie drobnego zabiegu na egzekutorze, przez wprowadzanie doń niewielkich modyfikacji, może stanowić istotną racjonalizację.

Powyższe zagadnienia były przedmiotem rozwiązań kilku ośrodków zetowskich. Znane są rozwiązania ZETO Lublin (patrz artykuł „Wynalazcy z ZETO Lublin", INFORMATYKA nr 5/79) oraz ZETO Łódź. W niniejszym artykule przedstawiamy rozwiązanie dokonane w ZETO Białystok dla egzekutorów E4BM, E6BM i E6RM, działających na komputerach ODRA serii 1300.

Prezentowane poniżej usprawnienia różnią się od pozostałych sposobem rozwiązania symulacji drukarki wierszowej. Zasadniczym ulepszeniem jest wprowadzenie blokowania rekordów w zbiorze ze składowanymi tabulogramami, co zapewnia optymalne wykorzystanie taśm magnetycznych (w praktyce stwierdzono około siedmiokrotnie mniejszą długość wykorzystanej taśmy magnetycznej niż w rozwiązaniu ZETO Lublin). Ponadto, możliwość jednoczesnego składowania wydruków z wielu programów pozwala na pełne korzystanie z wieloprogramowości. Pragniemy wspomnieć, że niniejsze rozwiązanie jest wynikiem kontynuacji podjętych już w 1974 roku w ZETO Białystok prac nad usprawnieniem egzekutora. Rozwiązanie to zostało już upowszechnione u szeregu użytkowników.

SYMULACJA DRUKARKI WIERSZOWEJ NA TAŚMIE MAGNETYCZNEJ

Idea rozwiązania problemu polega na zastąpieniu drukarki wierszowej programem EXECSYMULANT. Obsługę tego „urządzenia zewnętrznego" zapewnia zmodyfikowany egzekutor. Modyfikacje egzekutora polegają na zmianie obsługi ekstrakodów z grupy 15 (SUSBY, REL, DIS, ALLOT, PERI), odnoszących się do drukarki wierszowej. Dla programów symulowanych, tzn. programów, do których został przydzielony EXECSYMULANT, wszystkie ekstrakody (poza PERI) są ignorowane, po ewentualnym ustawieniu słów odpowiedzi. Natomiast na podstawie obszaru sterującego rozkazu PERI, wiersz wydruku przekazywany jest do programu EXECSYMULANT. Program ten zawiera operacje obsługi zbioru na taśmie magnetycznej za pomocą SAPu (System Automatycznego Programowania), a mianowicie: otwarcie i zamknięcie taśmy, zapis rekordu i zapis oznacznika pośredniego. Każda z tych operacji ma swój punkt wejścia, kończy się zaś specjalnym ekstrakodem (177), nielegalnym w oryginalnej wersji egzekutora.

Wykonanie tego ekstrakodu powoduje specjalne zawieszenie programu EXECSYMULANT. Rozkaz PERI z programu symulowanego powoduje, że zmodyfikowany egzekutor odwiesza EXECSYMULANTa, ustawiając odpowiedni punkt wejścia — w tym przypadku do zapisu rekordu. Wywołanie innych operacji i związane z tym odwieszenie programu dokonywane jest dodatkowymi komunikatami operatorskimi.

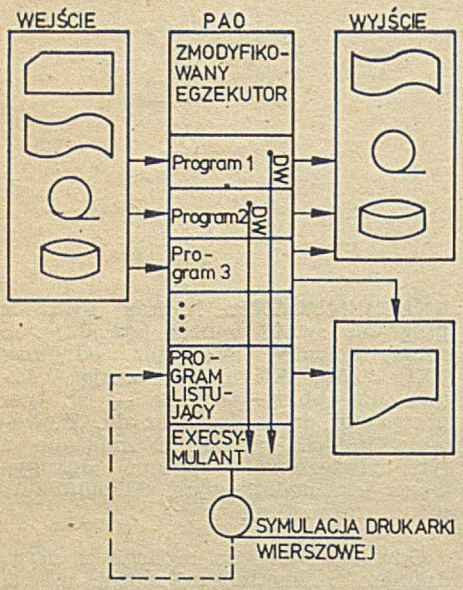
Odwołując się do analogii z urządzeniami zewnętrznymi można mówić o obciążeniu EXECSYMULANTa transmisją w trakcie wykonywania poszczególnych operacji i gotowości do przyjęcia transmisji w przypadku jego zawieszenia. Oczywiście, programy symulowane, żądające wyjścia na drukarkę podczas działania EXECSYMULANTa, muszą zostać zawieszone, oczekując na koniec transmisji i przejście w stan gotowości. Koordynację tych operacji zabezpiecza egzekutor z dokonanymi modyfikacjami.

Zapis rekordów w zbiorze symulacji odbywa się w kolejności zgłoszeń instrukcji PERI z programów symulowanych, z dowolną długością bloku (standardowo przyjęto 1024 słowa) w zależności od programu EXECSYMULANT. Każdy rekord oprócz licznika i wiersza wydruku zawiera pełną identyfikację, umożliwiającą odtwarzanie wydruków przez programy listujące. Identyfikacja tworzona jest w egzekutorze za pomocą wewnętrznej tabeli programów aktualnie symulowanych i zawiera numer wewnętrzny programu oraz programowy numer drukarki wierszowej z instrukcji PERI. Wydruk z każdego programu w zbiorze symulacji rozpoczyna się i kończy oznacznikiem pośrednim, zawierającym nazwę i numer wewnętrzny programu oraz czas rozpoczęcia lub zakończenia symulacji. Zapis oznaczników pośrednich następuje odpowiednio w momencie rozpoczęcia i zakończenia symulacji.

Listowanie zbioru symulacji może się odbywać na komputerach ODRA serii 1300 lub maszynach Jednolitego Systemu. Programy listujące dają możliwość:

— kompletnego listowania wydruków ze zbioru symulacji
— listowania wybiórczego
— listowania wydruków od wybranej strony.

Ogólny schemat przetwarzania z zastosowaniem symulacji drukarki wierszowej przedstawiony jest na rysunku 1.



Rys. 1. Ogólny schemat przetwarzania z zastosowaniem systemu symulacji

Zmodyfikowany egzekutor w porównaniu z wersją oryginalną jest o ok. 500 słów większy. Program EXECSYMULANT zajmuje 770 słów, nie licząc buforów wyjściowych dla zbioru symulacji. Program ten jest wprowadzany do pamięci operacyjnej za pomocą jednego z następujących komunikatów operatorskich:

LDTM — jeśli program znajduje się na taśmie magnetycznej lub dysku

LDTP — jeśli program znajduje się na taśmie papierowej.

Po wprowadzeniu do pamięci stają się dostępne następujące komunikaty sterujące procesem symulacji:

OP[EN] — otwarcie zbioru symulacji; powoduje otwarcie taśmy z przekroczonym okresem zachowania lub SCRATCH TAPE i zapisanie etykiety nagłówkowej SYMULACJA DW

DW#prog — przydzielenie EXECSYMULANTa do programu; powoduje rozpoczęcie symulacji wszystkich drukarek wierszowych danego programu na taśmie magnetycznej; komunikat ten może być użyty w dowolnym momencie dla wszystkich programów aktualnie znajdujących się w pamięci komputera

WD#prog — oddzielenie EXECSYMULANTa od programu — zakończenie symulacji; skasowanie programu z pamięci powoduje jego automatyczne oddzielenie

CL[OSE] — zamknięcie zbioru symulacji; powoduje zapisanie etykiety końca zbioru i zamknięcie taśmy po zakończeniu procesu symulacji

DL — skasowanie programu EXECSYMULANT z pamięci operacyjnej.

USPRAWNIENIE OPEROWANIA KOMPUTEREM

Rozwiązania tego problemu polegają na dokonaniu następujących modyfikacji:

— wprowadzeniu do egzekutora dodatkowych funkcji

— skróceniu lub całkowitej likwidacji niektórych zbędnych komunikatów wyprowadzanych przez egzekutor na monitor.

Poniższe specjalne komunikaty inicjują następujące dodatkowo wprowadzane funkcje egzekutora:

SC n — skasowanie taśmy magnetycznej w jednostce pamięci taśmowej o numerze operatorskim *n*

UL n — przydzielenie do systemu jednostki pamięci taśmowej po jej rozładowaniu, bez konieczności bezpośredniej manipulacji przyciskami przewijaka

PG#prog n m — przydzielenie do programu urządzenia *n* o numerze programowym Ø i start programu od punktu *m* (zastępuje dwa komunikaty GIVE i GO)

SI — blokowanie komunikatów egzekutora:
S#prog; UNIT q: — USED AS X
S#prog; UNIT q: — FREE

Powtórne SI odblokowuje wyprowadzenie wymienionych komunikatów.

Część komunikatów została usunięta na stałe, ponieważ blokują tylko monitor, nie dając operatorom istotnych informacji. Są to:
— komunikaty towarzyszące ładowaniu programu do pamięci
— komunikat S#prog; UNIT q: — — OPAT
— komunikat S#prog; CLOCKED: — — ab.

Pozostałe komunikaty zostały skrócone do takiej postaci, która zawiera minimalną ilość słów opisowych, a jednocześnie zapewnia pełną czytelność. Postać tych komunikatów przedstawia tabela 1.

Tabela. Postać komunikatów wyprowadzanych na monitor

Przed modyfikacją	Po modyfikacji
S#prog; UNIT q :- BUSY	S#prog; q:- BU
S#prog; UNIT q :- FIX	S#prog; q :- FX
S#prog; UNIT q :- FREE	S#prog; q :- FR
S#prog; UNIT q :- R&FX	S#prog; q :- R&F
S#prog; UNIT q :- RPTD	S#prog; q :- RPTD
S#prog; UNIT q :- Fail n	S#prog; q :- FAIL n
S#prog; UNIT q :- USED AS X	S#prog; q :- US X
S#prog; UNIT q :- OPAT	nie występuje
S#prog; CLOCKED :- ab	nie występuje
S#prog; ILLEGAL :- e ZZZZ	S#prog; IL :-e ZZZZ
S#prog; DELETED :- informacja	S#prog; DE :- informacja
S#prog; DISPLAY :- informacja	S#prog; DI :- informacja
S#prog; HALTED :- informacja	S#prog; HA :- informacja
CLOSED; UNIT q :- nazwa zbioru REPEATS n	CL; q :- nazwa zbioru R n
UNLOAD; UNIT q :- nazwa zbioru REPEATS n	UN; q :- nazwa zbioru R n
S#prog; LOAD nazwa zbioru	S#prog; LO nazwa zbioru
SCRATCH TAPE	SCRATCH
CANCEL	CAN
ERROR	ER
Komunikaty towarzyszące ładowaniu programu	nie występują

KORZYŚCI Z WPROWADZONYCH USPRAWNIEŃ

W wyniku przeprowadzonych zmian egzekutora uzyskano następujące efekty:

— uniezależnienie się od drukarki wierszowej; symulacja drukarki wierszowej umożliwia realizację programów wykorzystujących drukarkę w czasie, gdy jest ona zajęta lub niesprawna

— lepsze wykorzystanie wieloprogramowości, szczególnie w konfiguracji komputera z jedną drukarką wierszową

— lepsze wykorzystanie zasobów poprzez usprawnienie obsługi powtarzających się czynności operatorskich i skrócenie czasu oczekiwania na dostęp do monitora

— zmniejszenie awaryjności monitora (mniejsza liczba znaków wyprowadzanych na monitor powoduje mniejsze zużycie mechaniczne).

Z doświadczeń dotychczasowej eksploatacji zmodyfikowanego egzekutora wynika, że największe efekty przynosi zastosowanie go przy konfiguracjach komputerów z jedną drukarką wierszową, co najczęściej występuje w przypadku ODRY 1304. Ponadto przy wykorzystaniu wysokiego współczynnika wieloprogramowości celowe jest zastosowanie zmodyfikowanego egzekutora również dla konfiguracji ODRY 1305 z dwiema drukarkami wierszowymi.

Zmodyfikowany egzekutor z przedstawionymi w niniejszym artykule możliwościami został wdrożony w 11 krajowych ośrodkach obliczeniowych, spotykając się z pełnym uznaniem użytkowników.

Mgr Leonard DOBIECKI
inż. Zofia GURYN
ZETO Białystok

Po raz ostatni w Jeleniej Górze?

Trzecie Tenisowe Mistrzostwa Informatyków (sierpień 1980 r.) dowiodły, że impreza ta przestaje spełniać jedną z istotnych (przynajmniej dla jej sponsorów) funkcji, jaką jest integracja informatyków z różnych rejonów Polski. Uczestnicy jeleniogórskich Mistrzostw znali się już jak „łyse konie", zresztą nie tylko z racji uczestnictwa w rozgrywkach tenisowych, ale także np. narciarskich czy siatkarskich.

Formuła tej imprezy wymaga rewizji. W obecnej postaci impreza ma co prawda walor czynnego wypoczynku, ale ponieważ nie daje uczestnikom żadnych bonifikat, zachodzi obawa, iż w przyszłości zabraknie chętnych, którzy poświęciliby na nią część urlopu.

Należy więc przeprowadzać różnego rodzaju mistrzostwa, umożliwiając uczestnikom zawodów pobyt w towarzystwie rodzin lub bliskich. Odpowiednią formą mogłoby być, postulowane już przeze mnie, organizowanie rozgrywek przy okazji zwykłych urlopów pracowniczych — w miejscowości, która cieszy się dużą frekwencją informatyków. Kto tam będzie — skorzysta.

Tym razem mistrzostwa wygrała ponownie ekipa jeleniogórska, zdobywając na własność puchary. Następne zawody skłonny jest organizować Zakład koszaliński na kortach w Kołobrzegu. Zmiana miejsca i organizatora stwarza dodatkową szansę dla rewizji zasad imprezy.

(K.B.)

Złaz Informatyków

W dniach 16—21 września br. trwał coroczny Świętokrzyski Złaz Informatyków organizowany — siódmy już raz — przez Zakład Elektronicznej Techniki Obliczeniowej w Kielcach przy współpracy Zjednoczenia Informatyki i Zarządu Głównego Związku Zawodowego Energetyków.

Tegoroczny Złaz rozpoczął się w Kielcach, skąd uczestnicy wyruszyli do Wiślicy, a następnie wędrowali przez Skorocice, Chroberz, Krzyżanowice i Bogucice do Pińczowa. W Pińczowie odbyło się tradycyjne Ognisko Centralne. Przy ognisku zwycięzcy różnorodnych sprawdzianów i konkursów turystycznych przeprowadzanych na trasie Złazu otrzymali cenne nagrody (akcesoria turystyczne.

W VII Świętokrzyskim Złazie Informatyków wzięło udział blisko 50 informatyków z Zakładów Elektronicznej Techniki Obliczeniowej ze Szczecina, Jeleniej Góry, Krakowa, Gdyni, Zielonej Góry, Poznania, no i — oczywiście — Kielc.

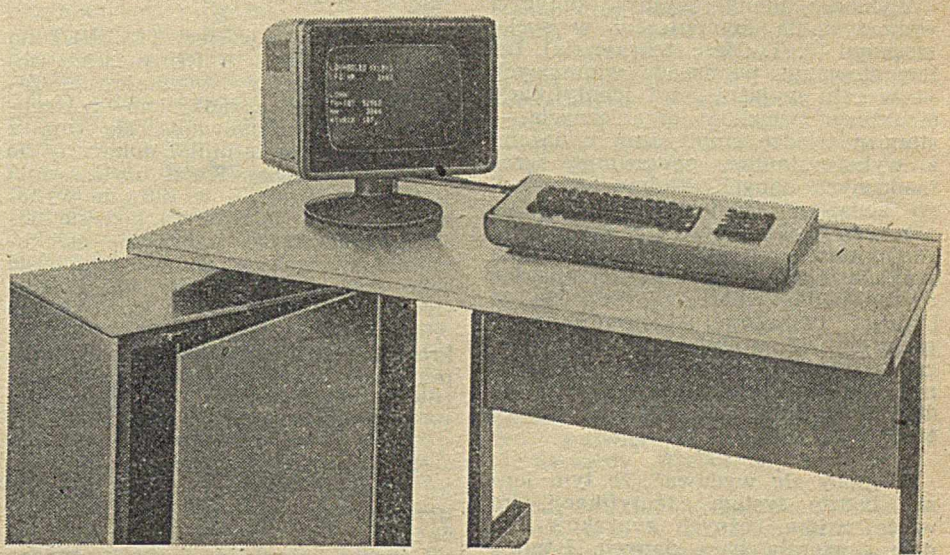
Celem Złazów jest wspólne wykorzystywanie kilku dni urlopu przez grupę informatyków-turystów, co służy integracji środowiska ZETO-wskiego. Ostatni Złaz — pod tym względem — spełnił swoje zadanie. **(I.K.)**

INTERBIURO '80

W ramach jesiennych salonów specjalistycznych Międzynarodowych Targów Poznańskich w dn. 12—17 października br. zorganizowany został (po raz trzeci) Międzynarodowy Salon Techniki Biurowej INTERBIURO. Kolejne Salony dzieli dwuletnia przerwa. Tym razem ekspozycja — zebrana w jednym pawilonie targowym — zawarła wyroby producentów z Czechosłowacji, NRD, Polski, RFN, Szwajcarii, Wielkiej Brytanii i Węgier. Oferta była znacznie uboższa niż w latach ubiegłych, zaś główny asortyment wyrobów (pod względem liczebności) stanowiły urządzenia powielające.

Środki techniki obliczeniowej, do tej pory zasadniczy element ekspozycji, przedstawiono w tym roku w bardzo zubożonej formie. Wprawdzie centrale VIDEOTON (WRL) i ROBOTRON (NRD) przygotowały się do Salonu solidnie i ich oferta była pokaźna, niemniej tylko do tych firm cgraniczyła się w praktyce prezentacja sprzętu informatycznego. Na wystawie niestety nie znalazł się ani jeden wyrób polskiego przemysłu komputerowego.

VIDEOTON przedstawił przede wszystkim: komputer R-11, służący do zdecentralizowanego przetwarzania informacji, terminal RPT 80, przeznaczony głównie do sterowania produkcją, dwa zestawy komputera biurowego VT 20, sterowanego mikroprocesorem i wyposażonego w pamięć dyskową, monitory alfanumeryczne serii VDT 52100, drukarki wierszowe oraz DISCOM — pamięć dyskową z dyskiem stałym.



robotron 4230

ROBOTRON zawarł w swojej ekspozycji: mały automat do fakturowania — robotron 1711, elektroniczny automat do księgowania — robotron 1355, automat do księgowania i fakturowania — robotron 1720 oraz system zbierania danych — robotron 4230 (fot.).

Oferta informatyczna była więc nadzwyczaj skromna i stanowiła dla środowiska olbrzymie zaskoczenie, zwłaszcza całkowitym zignorowaniem imprezy przez nasz przemysł. W opisanej sytuacji należałoby się zastanowić nad kontynuowaniem idei INTERBIURO. **(Z.G.)**

Informatyka w badaniach naukowych chemii

Międzynarodowa Szkoła Letnia „Data Processing in Chemistry", która odbyła się w dniach 26—31 sierpnia br. w Rzeszowie została zorganizowana przez Politechnikę Rzeszowską i Oddział Wojewódzki NOT, pod patronatem Komitetu Chemii PAN i Ministerstwa Przemysłu Chemicznego. W zajęciach szkoły wzięło udział ponad sto osób z kilkunastu krajów Europy oraz Japonii i USA. Trzynastu wykładowców (w tym trzech polskich) z ośrodków wiodących w skali światowej wypełniło pięciodniowy program, dostarczając uczestnikom wiele interesującego materiału związanego z zastosowaniami metod obliczeniowych w chemii. Tematyka zajęć objęła dwa kierunki badań: wykorzystanie metod spektroskopii i modelowanie reakcji chemicznych. Poszczególne wystąpienia zostaną tu scharakteryzowane pod względem specyfiki zagadnień obliczeniowych.

METODY OBLICZENIOWE W SPEKTROSKOPII

Wykład inauguracyjny dotyczył wykorzystania komputerów w spektroskopii (J. T. Clerc, Szwajcaria). Interpretowanie pomiarów widmowych — w celu wyjaśnienia i identyfikacji struktury związków organicznych — stanowi wiedzę empiryczną, związaną z wykorzystaniem poprzednio zgromadzonych danych. Widmo nieznanego związku otrzymane doświadczalnie jest porównywane ze wszystkimi elementami w bibliotece widm. Jeżeli zostanie stwierdzone dopasowanie dwóch widm, wówczas uważa się, że budowa związków reprezentowanych przez te widma jest identyczna. Z tego względu podstawowe znaczenie ma łatwy, komputerowy dostęp do dużych zbiorów danych spektrometrycznych.

Mogłoby się wydawać, że tym lepszy będzie system identyfikacji, im więcej widm różnych związków będzie zawierała baza danych. Istnieją tu jednak znaczne ograniczenia, np. zarejestrowanie widm 15 000 związków wymaga dysku o pojemności 100 megabajtów.

Efektywne przeszukiwanie bibliotek widm zapewnić może prosty, lecz dostatecznie szybki algorytm. Za najbardziej perspektywiczne pod tym względem uważa się kodowanie mieszające. Z obliczeniowego punktu widzenia, podjęte też zostały inne, ważne zagadnienia, a mianowicie: ocena podobieństwa widm, kodowanie struktury chemicznej związków i reguły decyzyjne.

Duże zbiory danych stanowią podstawę wielu technik automatycznej interpretacji widm, opartych zresztą nie tylko na metodzie przeszukiwania zbiorów, lecz także na metodach sztucznej inteligencji i rozpoznawania obrazów [1]. Liczba służących temu baz danych, które istnieją obecnie na świecie, nie przekracza dziesięciu. W Rzeszowie przedstawiono największą i najbardziej znaną — amerykański Chemiczny System Informatyczny (CIS, Chemical Information System [2]) instytucji NIH i EPA [3]) (S. R. Heller, USA).

System CIS składa się z kilkunastu baz danych, zawierających różnorodne informacje (m. in. widma) dotyczące wielu tysięcy związków chemicznych. Jego jądrem jest podsystem wyszukiwania struktur i nazw SANSS (Structure and Nomenclature Search System). Ponadto system ma dwie inne ważne cechy: stanowi katalog (złożony z kilkudziesięciu plików) będący rejestrem chemikaliów oraz umożliwia automatyczne opracowywanie danych przy użyciu wbudowanego zestawu programów.

System działa w sieci TELENET na komputerze PDP-10 firmy DEC. Dostęp jest możliwy telefonicznie, zatem jedynym urządzeniem potrzebnym do komunikowania się (np. z Warszawy) jest terminal dołączony do sieci telefonicznej. Koszt użytkowania jest zróżnicowany i w Europie wynosi ok. 30—40 dolarów za godzinę dostępu (plus koszt połączenia telekomunikacyjnego). W połowie bieżącego roku system użytkowało 350 zarejestrowanych organizacji i wielokrotnie więcej indywidualnych osób z 40 krajów. Zdaniem autorów, nawet przy tak znacznym obciążeniu, możliwości instalacji są wykorzystywane tylko w połowie.

Główną cechą systemu jest nie tylko łatwość dostępu, ale i prostota użycia. Przy jego tworzeniu założono, że chemik ma zwykle wiele informacji prowadzących do automatycznego rozwiązania postawionego zadania, lecz nie zna najlepszych pytań, jakie należy zadać komputerowi. Stąd też zwykły tryb interakcyjny, polegający na zadawaniu pytań komputerowi, został tu częściowo odwrócony — komputer także pyta użytkownika, np.: „ostatnie kryterium spełnia 17 450 związków, czy chcesz zmienić kryterium?". W ten sposób użytkownik jest wykorzystywany jako „przetwornik" informacji, co dzięki dwukierunkowemu sprzężeniu z komputerem znacznie przyspiesza rozwiązywanie zadań. Narzuca to zupełnie odmienną filozofię oprogramowania systemu.

Z natury rzeczy nie istnieje tu problem ochrony pamięci, zarówno przy odczycie (cała informacja jest dostępna dla wszystkich) jak i zapisie. Dołączenie nowych elementów do systemu wymaga sprawdzenia ich przydatności w systemie pilotowym i trwa ok. jednego roku. Prawo zapisu ma tylko jedna osoba.

Drugie podejście do zagadnień identyfikacji związków chemicznych, które można nazwać strukturalnym, reprezentowane było przez dwóch autorów (L. A. Gribow, ZSRR; S. Sasaki, Japonia). Wyróżniającą cechą tej grupy metod jest wykorzystanie sztucznej inteligencji, a najbardziej znaną (również informatykom) realizacją jest amerykański rozbudowany program DENDRAL [4]).

W tym wypadku identyfikacja związku odbywa się w sposób podobny jak w metodzie poprzedniej, tzn. przez porównanie widma otrzymanego doświadczalnie z widmem wzorcowym. Różnica polega zaś na tym, że widmo wzorcowe jest generowane w procesie identyfikacji przez komputer — na podstawie początkowych hipotez dotyczących budowy cząsteczki, ze stopniowym wprowadzaniem poprawek do znalezionej struktury. Istotą metody jest zatem konstruowanie widma teoretycznego i porównywanie go z doświadczalnym (w celu określenia ich podobieństwa), co pozwala na określenie budowy związku.

Pod względem metodologii wszystkie programy tej grupy, tj. STREC [5]), CHEMICS [6]) i DENDRAL nie różnią

[1]) J. T. Clerc, p. 393, Computers in Chemical Education and Research Plenum, New York, 1977

[2]) S. I. Heller, B. W. A. Milne, R. J. Feldmann, Science, 195, 253 (1977)

[3]) NIH — National Institutes of Health, EPA — Environmental Protection Agency, rządowe instytucje USA zajmujące się problematyką ochrony zdrowia oraz ochrony środowiska

[4]) B. B. Buchanan, E. A. Feigenbaum, Artificial Intelligence, 11, 5 (1978)

[5]) L. A. Gribow, Analitica Chimica Acta (Computer Techniques and Optimization), 95, 75 (1977)

[6]) S. Sasaki et al., Journal of Chemical Information and Computer Sciences, 18, 211 (1973)

się między sobą. Podstawowe algorytmy procedury są określone przez kolejne etapy procesu identyfikacji, na które składają się:

- analiza zgodności znanych wcześniej elementów strukturalnych (tj. wybranych atomów lub cząsteczek) z danymi doświadczalnymi, której konsekwencją jest wyeliminowanie już na wstępie możliwie największej liczby hipotez najmniej prawdopodobnych

- generowanie kompletnych struktur z elementów strukturalnych, co jest równoznaczne z utworzeniem przestrzeni poszukiwań; w ten sposób uzyskuje się zestaw kandydatur tworzących dopuszczalny obszar poszukiwania rozwiązania

- porównywanie przewidywanych teoretycznie właściwości nieznanego związku (widmowych, strukturalnych) z właściwościami rzeczywistymi określonymi doświadczalnie, w celu znalezienia struktur najbardziej podobnych.

Procedura taka musi być powtarzalna i wykonywana interakcyjnie, a jednocześnie łatwa do opanowania dla użytkownika — bez specjalnego przygotowania informatycznego.

Niewątpliwie najważniejsze i najbardziej złożone spośród algorytmów, to algorytmy generacji struktur. Są to procedury heurystyczne polegające na komputerowej symulacji procesu decyzyjnego, jaki wykonuje chemik, aby zinterpretować dane doświadczalne (widmowe). W ten sposób działa generator CONGEN, będący sercem zespołu programów znanych pod nazwą DENDRAL. Jest to program napisany w języku INTERLISP, tak rozbudowany, że nie został dotąd w całości nigdzie przeniesiony i autorzy zalecają jego użytkowanie wyłącznie na sprzęcie dostępnym w miejscu jego powstania, tj. w Uniwersytecie w Stanford. Jednakże, nie bacząc na trudności rozpoczęto adaptację programu CONGEN do systemu CIS, co jest wyrazem pożądanej tendencji i — jak twierdzą eksperci — nieuchronnej konieczności połączenia się obu dróg wyjaśniania budowy związków chemicznych.

MODELOWANIE REAKCJI CHEMICZNYCH

Modelowanie reakcji organicznych polega głównie na modelowaniu ich struktury. Synteza organiczna ma podstawowe znaczenie przy produkcji nowych związków chemicznych, takich jak np.: leki, paliwa syntetyczne, tworzywa sztuczne. Wykrycie właściwej drogi syntezy przy istnieniu kilku milionów dotychczas zsyntetyzowanych związków i kilkudziesięciu tysięcy znanych reakcji jest niezwykle trudne bez pomocy komputera. Z tego względu podstawowe znaczenie mają programy komputerowe pomagające chemikom w projektowaniu syntezy złożonych cząsteczek metodą heurystycznego przeszukiwania przestrzeni właściwych dróg syntezy. Celem nie jest oczywiście zastępowanie pracy chemika, lecz wyposażenie go w wygodne narzędzie.

Zagadnienia komputerowej syntezy związków organicznych [7]) zostały omówione przeglądowo (Z. Hippe, Polska).

Istnieją dwa uzupełniające się podejścia do zagadnień komputerowego rozwiązywania problemów syntezy chemicznej. Pierwsze — zwane kojarzącym bezpośrednio (ang. *direct-associative)* — wymaga znajomości substratów i reakcji prowadzących do otrzymania produktu. Drugie — zwane ukierunkowanym logicznie (ang. *logic-centred)* — polega na analizie struktury związku docelowego i określeniu możliwych sposobów jej powstania, a następnie na przechodzeniu wstecz od struktury docelowej do początkowej (prostszej) i poszukiwaniu w każdym pojedynczym kroku struktur pośrednich. Powtarzanie tej procedury w każdym kroku, aż do uzyskania łatwo dostępnego związku, prowadzi do wygenerowania drzewa struktury. W praktyce chemik przedstawia pożądaną budowę cząsteczki, a komputer generuje możliwe drogi syntezy, z których chemik w sposób interakcyjny wybiera niektóre, podejmując także decyzje, kiedy zakończyć proces poszukiwań.

Fundamentalne znaczenie w tej metodzie ma grafika komputerowa. Przykładowo, program SECS [8]), którego autor niestety nie przybył do Rzeszowa, wymaga intensywnego korzystania z terminala graficznego (np. GT-40 dołączonego do komputera PDP-10) podczas całego procesu projektowania syntezy. W praktyce chemik rysuje docelową strukturę i uczestniczy w jej przetwarzaniu posługując się piórem świetlnym, choć do użytkowania systemu SECS wystarcza dostęp do sieci telefonicznej i terminala.

Informatyków zainteresować może fakt, że w programie SECS reakcje nie są reprezentowane przez podprogramy. Zamiast tego wykorzystuje się swego rodzaju „reguły produkcji", stanowiące bardziej naturalny odpowiednik reakcji chemicznych. Wynikło to z konieczności rozdzielenia strategii heurystycznych poszukiwania dróg syntezy od samych reakcji, które należało traktować jako rozszerzalny zbiór danych, tzn. z możliwością dołączenia opisu nowo odkrytych reakcji. Dlatego transformacje nie mają postaci programów, lecz stanowią zbiór faktów, w szczególności nie zawierają instrukcji skoku, pętli, ani wywoływania podprogramu. Konsekwencją tej sytuacji jest konieczność opracowania maksymalnie uproszczonego języka konwersacyjnego, który byłby łatwo zrozumiały dla chemika i odpowiadał jego aparatowi pojęciowemu.

Czynnikiem warunkującym skuteczność tej metody, a jednocześnie uzależniającym ją od zasobów komputerowych i wiedzy jej twórców jest dostępność kartoteki znanych reakcji chemicznych. Cechę tę wyeliminowano w innym wariancie omawianego podejścia, reprezentowanym na zajęciach Szkoły przez referaty dwóch autorów (I. Ugi, J. Gasteiger, RFN). Zamiast zbioru danych wykorzystuje się tu model matematyczny zjawisk chemicznych, w którym cząsteczki i reakcje przedstawione są przez macierze. W szczególności umożliwia to reprezentację struktur cząsteczkowych w komputerze i wygodną nimi manipulację. Macierze mają szereg interesujących właściwości matematycznych (z interpretacją geometryczną), z miarą „odległości chemicznej" włącznie. Minimalizacja tych odległości jest pożądana ze względu na skracanie drogi syntezy. Stosuje się tu ogólne algorytmy optymalizacji, wykorzystywane także w innych dziedzinach nauki.

Ambicją wymienionych autorów jest opracowanie programu, który działa bez jakiejkolwiek interakcji z osobą użytkującą go — w oparciu o wstępnie przekazywaną mu informację. Dzięki temu unika się wielu kłopotów związanych z uczestnictwem chemika w procesie poszukiwania dróg syntezy. Jednakże rezygnacja z wpływu specjalisty na proces poszukiwań wymaga podania pełnej informacji o właściwościach chemicznych już na etapie tworzenia modelu i formułowania heurystyk. Zaprojektowany w ten sposób program EROS [9]) generuje drzewa syntezy poczynając od obiektu docelowego bądź początkowego, a łącząc w sobie oba (wymienione na początku tego punktu) podejścia do zagadnień syntezy komputerowej reakcji chemicznych.

Olbrzymie znaczenie we wszystkich zastosowaniach generacji struktur mają bardzo atrakcyjne i skuteczne metody teorii grafów [10]), którym poświęcono oddzielny wykład (J. Kennedy, Wielka Brytania).

Podejście teoretyczne do problemów chemii było w ostatnich dziesięcioleciach kształtowane przez mechanikę kwantową, należy się zatem liczyć z dalszym wykorzystywaniem jej dorobku. Zagadnieniom obliczeniowym w tej niezwykle ważnej dziedzinie poświęcone było oddzielne wystąpienie (A. Sadlej, Polska).

---

[7]) Z. Hippe, R. Hippe, p. 172, Proc. 4th International Symposium System-Modelling-Control, Zakopane, 1979

[8]) W. T. Wipke et al., Artifical Intelligence, 11, 173 (1978)

[9]) J. Brandt et al., p. 33, Computer-Assisted Organic Sybthesis, American Chemical Society, Washington, 1977

[10]) R. E. Tarjan, p. 1, Algorithms for Chemical Computations, American Chemical Society, Washington, 1977

W Polsce, przy bardzo ograniczonych możliwościach sprzętowych, perspektywę ma rozwój własnych metod obliczeniowych, we współpracy z ośrodkami zagranicznymi z dostosowaniem do naszych potrzeb [11]). Osiągnięcia ośrodka rzeszowskiego w tej dziedzinie przedstawiono w oddzielnym wykładzie i — bardzo szeroko — na tzw. sesji plakatowej (ang. *poster session)* (Z. Hippe i współpracownicy).

Zagadnienia modelowania reakcji chemicznych w sposób ilościowy (tj. obliczanie przebiegów i rozkładów wielkości fizycznych), z pewnością nie obce informatykom, polegają zazwyczaj na rozwiązywaniu układów równań różniczkowych i algebraicznych. Dotychczas opracowano wiele programów komputerowych przeznaczonych do rozwiązywania zagadnień tego rodzaju, a jeden z nich — CRAMS [12]) przedstawiono na zajęciach Szkoły (P. deMaine, USA). Cechą wyróżniającą ten program ze wszystkich innych tego typu jest łączne ujęcie zagadnień obliczania wartości zmiennych, tj. stężeń reagentów (problem symulacji), oraz wartości parametrów, a więc stałych szybkości reakcji i równowagi (problem dopasowania krzywych). Wynikło to z istoty problemów, przed jakimi stoją chemicy, i z postaci danych, jakimi dysponują.

Program CRAMS umożliwia rozwiązywanie dowolnie skomplikowanego układu równań kinetyki i równowagi przy dowolnej kombinacji wymienionych danych. Poza wykonywaniem obliczeń program ma unikalną możliwość pracy w tzw. reżimie predykcji, który polega na określaniu różnych kombinacji parametrów niezbędnych do obliczania wartości pozostałych wielkości nieznanych. Ważną cechą programu jest także możliwość uwiarygodnienia modelu, polegająca na sprawdzeniu, czy układ równań stanowi wiarygodny opis, odpowiadający danym doświadczalnym (których również używa się do testowania).

W przeciwieństwie do innych programów tego typu system CRAMS ma bardzo wyspecjalizowany moduł współpracy z użytkownikiem. Do tego celu stworzono specjalny język ukierunkowany na typowe problemy spotykane w chemii.

* * *

Pod względem naukowym Szkoła była zorganizowana znakomicie i stała na bardzo wysokim poziomie, jako że wykładowcy stanowili czołówkę światową w tej dziedzinie. Można mieć wszak pewne zastrzeżenia do kompletności przedstawienia niektórych zagadnień lub rozbudowania jednych kosztem innych.

Zabrakło — na przykład — omówienia niezwykle płodnej, wyłonionej w ostatnich latach dziedziny zwanej chemometrią [13]), która dotyczy chemicznych zastosowań takich metod matematycznych, jak: rozpoznawanie obrazów, analiza czynnikowa, teoria optymalizacji i in. Jednakże mimo wielkiej wiedzy i doświadczenia niewiele można powiedzieć na temat metody rozpoznawania obrazów w chemii, jeżeli ma się do dyspozycji 45 minut, a w dodatku dzieli ten czas na omówienie techniki identyfikacji obiektów (Z. Bubnicki, Polska).

Pewnym zaskoczeniem może być także fakt, że w czasie obrad zaledwie na jednym wykładzie padło słowo **mikrokomputer** (R. A. Spraggs, Wielka Brytania), mimo że obecnie mikrokomputery mogą stanowić rozsądną alternatywę nawet w obliczeniach teoretycznych [14]). Podobnie rzecz się miała z zagadnieniami automatyzacji eksperymentów (E. Steger, NRD).

Niemniej, jestem zdania, że skupienie się na dwóch, w gruncie rzeczy wąskich kierunkach, dało pozytywny efekt, skonsolidowało uczestników. Przy rozszerzeniu tematyki Szkoła prawdopodobnie nie miałaby tak zdecydowanego profilu, co mogłoby się odbić niekorzystnie na jej wynikach.

Tym razem podczas dyskusji, czuło się, że widownia ma swą własną tożsamość.

Uczestnicy orzekli, że zarówno w spektroskopii, jak i w modelowaniu reakcji chemicznych największą przyszłość mają metody sztucznej inteligencji i rozpoznawania obrazów. Spośród głosów podsumowujących problematykę wymienionych kierunków warto przytoczyć stwierdzenie, że (Z. Kęcki, Polska): *W obu przypadkach musimy korzystać ze zbioru danych, jednak powinniśmy robić to z myślą o rozwoju teorii.*

Natomiast, z obrad nie bardzo wynikało, czego chemicy oczekują od informatyków. Można się domyślać, że w warunkach polskich — ze względu na praktyczną niemożność zakupu odpowiedniego sprzętu; zarówno w kraju, jak i za granicą — chemicy stawiają na rozwój algorytmów obliczeniowych, w czym informatycy nie zawsze mogą im pomóc. Z drugiej strony informatyzacja badań naukowych staje się tak powszechna i coraz łatwiej dostępna, że chemicy są w stanie sami rozwiązywać swoje problemy.

Pod względem organizacji przebieg Szkoły należy uznać za bardzo udany. Pomijając sytuacje niezależne od organizatorów (np. brak kilku zapowiedzianych wykładowców), nie było powodów do narzekań.

Janusz ZALEWSKI

[11]) Z. Hippe, red., Applications of Computers in Processing of Physicochemical and Analytical Data, Wydawnictwa Politechniki Rzeszowskiej, 1979

[12]) P. A.D. deMaine et al., p. 25, Computers in Chemical Education and Research, Plenum, New York, 1977

[13]) B. R. Kowalski, red. Chemometrics: Theory and Applications, American Chemical Society, Washington, 1977

[14]) N. S. Ostlund, Intern. Journal of Quantum Chemistry: Quantum Chemistry Symposium, 13, 15 (1979)

Po INFOGRYFIE

Jak już zapowiadaliśmy, w dniach 17—20 listopada br. odbyły się przełożone z września V Kołobrzeskie Dni Informatyki INFOGRYF'80. Tegoroczny program obrad konferencji koncentrował się w czterech sekcjach problemowych (projektowanie systemów, konstrukcja oprogramowania, zastosowania informatyki w zarządzaniu oraz informatyki w dydaktyce). Zgodnie z tradycją odbyły się również Giełda Systemów oraz Sesja Studencka, która oprócz problematyki studenckiego ruchu naukowego, objęła dyskusję „okrągłego stołu" pt. „Informatyk — zawód przyszłości czy stracone złudzenia". Drugą, ogólną dyskusję „okrągłego stołu" zatytułowano: „Informatyk — zawód czy hobby".

Szczegółową relację z przebiegu imprezy oraz najciekawsze z wygłoszonych referatów postaramy się opublikować w przyszłorocznych numerach INFORMATYKI. (Z)

ZWIĄZKI ZAWODOWE

W środowisku informatyków działa aktualnie kilka związków zawodowych (patrz: Komunikat Sekretariatu Komitetu Informatyki, INFORMATYKA nr 10/80, s. 33). Sytuacja jest o tyle niejasna, że przy jednakowych zadeklarowanych celach działalności — różne są metody i cele rzeczywiste. Chcielibyśmy wyjaśnić obecny stan, ułatwiając informatykom zorientowanie się w sytuacji. Z tą myślą publikujemy tekst członków NSZZ „Solidarność", zamieszczony pierwotnie w Informatorze Wewnętrznym dla Kół Nauki, Oświaty i Techniki NSZZ „Solidarność" region Mazowsze. Czekamy na następne głosy. (Red.)

Niezależni i samorządni

Wszyscy wiemy, jak traktować fakt przekształcania się starych, CRZZ-owskich związków w niezależne i samorządne organizacje. Wątpliwości mogą nasuwać się wówczas, gdy mamy do czynienia ze związkami nowo powstającymi. Tworzenie przynajmniej niektórych z nich prowadzi do rozbicia prawdziwie niezależnego i samorządnego ruchu związkowego, rozwijającego się na mocy podpisanego w Gdańsku Porozumienia. Oto krótka historia narodzin takiego związku — NSZZ Pracowników Informatyki w Polsce z siedzibą w Warszawie. Powstał on w Centrum Projektowania i Zastosowań Informatyki (CPiZI) w Warszawie.

W pierwszych dniach września stara Rada Zakładowa ZZ Energetyków w CPiZI przeprowadziła wśród pracowników ankietę, zawierającą dwa pytania: czy chcesz Związku Informatyków i — jeśli tak — czy ma on wchodzić do CRZZ. Oczywiście wszyscy chcieli niezależnych i samorządnych związków poza CRZZ. Jednocześnie na zebraniu załogi (4 września br.) grupa pracowników ogłosiła informację o powstaniu Komitetu Założycielskiego NSZZ — tego, który potem przekształcił się w „Mazowsze". Nastąpiła natychmiastowa reakcja starej Rady: powołując się na wyniki ankiety, wmawiając zebranym, że specyfika ich zawodu nie pozwala na przyłączenie się do związków terytorialnych i dokumentując potrzebę odrębności złymi doświadczeniami z ZZ Energetyków — doprowadzono do wyboru Komisji, która miała opracować statut nowego związku branżowego.

W skład Komisji nie weszli oczywiście członkowie KZ NSZZ „Mazowsze", weszli natomiast: pomysłodawca — były dyrektor w OBRI, obecnie kierownik zakładu (członek egzekutywy POP), prowadzący to zebranie kierownik działu (bezpośrednio podległy dyrekcji), jeszcze jeden kierownik zakładu, przewodniczący starej rady zakładowej oraz bardzo popularny wśród załogi kolega, od dłuższego czasu związany z niezależnym ruchem społecznym; ten ostatni po pewnym czasie wystąpił z Komisji. Początkowo Komisja grała na zwłokę, ale zmuszona do działania akcją prowadzoną wśród pracowników przez KZ NSZZ „Mazowsze", opracowała w ciągu dwóch tygodni statut. Odpowiadał on bardziej statutowi stowarzyszenia niż związku (np. punkt o dwóch członkach wprowadzających).

Po zrealizowaniu celu, dla którego została powołana (tj. po opracowaniu statutu) Komisja nie rozwiązała się, lecz przemianowała na Komitet Założycielski NSZZ Pracowników Informatyki. Powołując się na wyimaginowany mandat załogi, prowadziła ostrą kampanię przeciwko NSZZ „Mazowsze". Na przykład jeden z jej członków zebrał swoich pracowników i straszył siłami antysocjalistycznymi, KOR-em itp. Twierdzono, że tylko związek branżowy może skutecznie bronić interesów pracowniczych. Korzystając z poparcia dyrektora (telex, delegacje itp.) oraz działając poprzez stare Rady Zakładowe i dyrekcje innych zakładów, komisja rozszerzyła swą aktywność na całą Polskę (głównie na Zjednoczenie Informatyki). W efekcie doszło do zwołania zjazdu ogólnopolskiego i przedstawienia do rejestracji nieco już zmienionego statutu.

Powstał więc NSZZ Pracowników Informatyki w Polsce. Przewodniczącym Zarządu Głównego tego związku oraz Rady Zakładowej w CPiZI został były przewodniczący RZ Energetyków w tejże instytucji. Związek został zarejestrowany bez kłopotu; odpowiednie ogłoszenia ukazały się w prasie i TV; rozpoczęła się działalność na szerszą skalę (pisma do Ministerstwa, Zjednoczenia, Sejmu), mająca stworzyć wrażenie, że związek reprezentuje ogół tej grupy zawodowej. W rzeczywistości zrzesza on nie więcej niż 1000 członków, podczas gdy w Zjednoczeniu Informatyki pracuje 6500 osób (51% zrzeszonych w „Solidarności"); ogółem jest w kraju 60 tys. informatyków. W CPiZI — przy 700-osobowej załodze — związek zrzesza ok. 150 osób (stan z 24.10. 1980 r.).

Nie sądzimy, by większość członków tego związku zdawała sobie sprawę z jego „rozbijackiego" charakteru; nie wynika on przecież ze statutu, ale ze sposobu działania; sposobu, który czasem określa się jako działanie „ze starego portfela".

Włodzimierz GRUDZIŃSKI
Ryszard KASPRZYK
Centrum Projektowania i Zastosowań Informatyki

Postulaty

Czym jest informatyka w polskich realiach, a czym mogłaby być, gdyby nie... Gdyby nie co? Jakie decyzje, jakie fakty z niedawnej historii wpłynęły ujemnie na jej rozwój? Co trzeba by zmienić, co naprawić? Jakimi metodami należałoby działać?

Gorąca prośba! Piszcie do nas. Oczekujemy zwięzłych postulatów dotyczących organizacji informatyki w Polsce. Efekty takiej korespondencyjnej dyskusji przedstawimy na naszych łamach.

REDAKCJA

Związek czy Stowarzyszenie ?

Praca informatyków ma charakter usługowy, a jednocześnie jest ona — z racji specyfiki wykorzystywanej przez nich technologii — w pewnym stopniu uniezależniona od jednostki organizacyjnej, na rzecz której działają. Informatycy, mimo że rozproszeni po całym kraju (w różnych zjednoczeniach, przedsiębiorstwach, zakładach) stanowią jednak zwartą, wyodrębnioną grupę. Świadomość wyjątkowej roli swojego zawodu dopingowała ich do konstytuowania organizacji reprezentującej środowisko. Przez kilkanaście lat czyniono próby powołania ogólnopolskiego Stowarzyszenia; wszystkie jednak spełzły na niczym. Jedynym osiągnięciem było (jak już informowaliśmy) utworzenie w styczniu br. lokalnego, Łódzkiego Stowarzyszenia Informatyków.

Obecnie (po sierpniu) powołanie organizacji reprezentującej całe środowisko stało się realne. Zarejestrowano już Niezależny Samorządny Związek Zawodowy Pracowników Informatyki (NSZZPI), który ma prawo — między innymi — powoływać związki twórcze i stowarzyszenia. Nowa sytuacja napawa optymizmem, daje szanse uporządkowania informatycznych spraw. Pojawiły się też jednak wątpliwości, które (z pomocą Czytelników) chcielibyśmy wyjaśnić.

Branżowy związek zawodowy może być reprezentantem jedynie większych skupisk informatyków (głównie pracowników ZETO). Wszyscy pozostali wybiorą zapewne związki zawodowe działające w przedsiębiorstwach macierzystych, tj. w tych, które ich zatrudniają. Poza tym istniejący już NSZZPI nie należy do NSZZ „Solidarność", co dla wielu osób jest powodem rezygnacji z przystąpienia doń. Tak czy inaczej — trudno przypuszczać, by jakikolwiek związek zawodowy mógł w przyszłości uznać siebie za reprezentanta całego środowiska.

Informatyków wiążą bardziej racje intelektualne i zawodowe, niż socjalne, którymi przede wszystkim zajmują się Związki (patrz: Uchwała I Zjazdu NSZZPI, INFORMATYKA nr 11/80, str. 29). Dlatego też — jak można sądzić — jeszcze ważniejsze jest dla środowiska powołanie organizacji o charakterze ściśle zawodowym, tj. Stowarzyszenia. Powinno ono zostać ukonstytuowane niejako oddolnie, niezależnie od instytucji zarządzających informatyką. Być może udałoby się wykorzystać uprawnienia związków zawodowych i w oparciu o nie powołać Stowarzyszenie, które miałoby w efekcie większe szanse zintegrowania środowiska.

Wszystko to piszemy ze sporą dozą niepewności, sytuacja nie jest wszak jednoznaczna. Co potrzebne jest bardziej — Związki czy Stowarzyszenie? Na jakich zasadach określone? W jakim kształcie ukonstytuowane? Liczymy bardzo na Wasze głosy, liczymy, iż pomożecie nam w opisie i ocenie aktualnej, tak dynamicznie zmieniającej się sytuacji. **(Red.)**

LISTY

Szanowny Panie Redaktorze!

W artykule pt. „POSTULATY 1980" (INFORMATYKA nr 10/80) Władysław Łada czyni Głównemu Urzędowi Statystycznemu wymówkę z powodu publikowania informacji o wykorzystaniu czasu pracy komputerów, widząc w tym jeden z antybodźców racjonalnego wykorzystania tego sprzętu. Sądzę, że nie jest to zarzut uzasadniony, ponieważ bez badania czasu pracy nie mielibyśmy żadnych podstaw do ogólnej, choćby nawet niepełnej oceny wykorzystania komputerów. Oprócz wskaźników wykorzystania komputerów, wyprowadzanych z bilansu czasu pracy, bardzo potrzebne byłyby wskaźniki efektywności ich wykorzystania, podobnie jak oprócz wskaźników wykorzystania czasu pracy robotników są niezbędne wskaźniki wydajności ich pracy.

Wspomniany przez autora iloraz wartości sprzedaży usług przez liczbę zatrudnionych, którego zresztą w publikacji GUS (OBR SPIS) się nie wyprowadza, nie jest żadną miarą efektywności wykorzystania komputerów.

Ośrodek Badawczo-Rozwojowy SPIS przewidział w swym planie na 1981 r. próbę znalezienia metody mierzenia efektywności wykorzystania komputerów, którą można by było zastosować w statystycznych badaniach ośrodków informatyki. Byłbym zobowiązany Czytelnikom INFORMATYKI za propozycje dotyczące takiej metody. Mam przy tym nadzieję, że również autor wspomnianego artykułu zechce takowe przekazać.

Dr Jan ISZKOWSKI
dyrektor OBR SPIS

Do Redakcji!

Niniejsze sprostowanie jest wynikiem błędów popełnionych przeze mnie. Końcowa postać artykułu pt. „Systemy Informatyczne w Kombinacie Górniczo-Hutniczym Miedzi", którą redagowałem w oparciu o materiały informacyjne przekazane przez współautora, Tomasza Tymeńskiego, nie była przez niego autoryzowana. Dokonane skróty wypaczyły część informacji i w rezultacie artykuł nie odzwierciedla faktycznego stanu informatyki w KGHM. Ze względu na reperkusje artykułu w środowisku informatyków i użytkowników systemów w Kombinacie prostuję następujące sprawy:

- *system SEPAR został w całości opracowany i wdrożony przez Instytut Górnictwa i Centrum Obliczeniowe Politechniki Wrocławskiej, przy współpracy zakładów górniczych KGHM*
- *zaniechano realizacji systemu SZLAK*
- *obliczanie płac nie obejmuje wszystkich zatrudnionych*
- *system Kontroli Ruchu Załogi w ZG Polkowice został wycofany z eksploatacji, ze względu na brak środków dewizowych na jego utrzymanie*
- *wrażenie wynikające z treści artykułu, jakoby przewidywano budowę sieci komputerowej przy wykorzystaniu obecnego sprzętu jest niesłuszne.*

Waldemar ZAGRAJEK

Szanse i granice przetwarzania informacji

Interesujący artykuł Władysława Klepacza zamieszczony w nr. 7-8/1980 INFORMATYKI, dotyczący VI Międzynarodowego Kongresu Informatyki Europejskiej w Wiedniu (który odbył się w marcu br.), ujmujący w zwięzłej i syntetycznej formie zarówno merytoryczną treść obrad, jak i panującą na nim doskonałą atmosferę, pozostawił we mnie — również uczestnika tego Kongresu — pewien niedosyt. Jest rzeczą zupełnie zrozumiałą, że nie sposób w jednym krótkim artykule zrelacjonować bogactwa tematyki zawartej w 112 wygłoszonych na Kongresie referatach, tym bardziej, że pojedynczy uczestnik mógł wysłuchać (wynikało to z harmonogramu obrad) co najwyżej dwudziestu kilku.

Sądzę, że nie będzie przesadą powrócić jeszcze raz do tematyki VI Międzynarodowego Kongresu w Wiedniu poniekąd i dlatego, że jako przedstawiciel Zjednoczenia Informatyki byłem zainteresowany bardziej użytkową problematyką informatyki, również bogato na Kongresie prezentowaną.

Kongres posiadający od lat tradycyjną nazwę „Przetwarzanie danych w Europie" przebiegał pod dodatkowo sformułowanym hasłem „Szanse i granice przetwarzania informacji". W znakomitej większości referatów, jak i w kończącej Kongres dyskusji panelowej, powyższy przewodni motyw był naświetlany w kontekście wielu czynników kształtujących aktualne trendy rozwojowe technik informatycznych oraz wynikające stąd implikacje społeczne, socjologiczne czy prawne. Jak widziane były „szanse i granice" w zakresie rozwoju sprzętu i oprogramowania, a więc w dwóch płaszczyznach, z którymi w codziennej praktyce zawodowej spotykamy się najczęściej? Temu tematowi postanowiłem poświęcić niniejszy artykuł opracowany na podstawie 22 referatów, których zdołałem wysłuchać podczas obrad Kongresu.

SPRZĘT INFORMATYCZNY

Odnosząc aktualny stan konstrukcji jednostek centralnych do stanu konstrukcji z 1950 r. można dokonać następującego porównania: konstrukcja obecna o tych samych gabarytach i koszcie, co w 1950 r., jest milion razy szybsza i posiada o jeden milion większą pamięć operacyjną; przyjmując zaś jednakową wydajność obliczeniową obu konstrukcji — obecna jest milion razy mniejsza gabarytowo i milion razy tańsza [2].

Dawny, automatycznie pracujący komputer w niewielkiej konfiguracji urządzeń zewnętrznych zastąpiony został przez system wyposażony w różnorodną liczbę urządzeń realizujących funkcje wejścia, pamięci i wyjścia, przy zastosowaniu wyspecjalizowanych technologii przetwarzania, a ponadto powiązany teledacyjnie z innymi komputerami, tworzącymi sieć komputerową.

Aktualnie w krajach rozwiniętych nie realizuje się już w zasadzie nowych systemów przetwarzania informacji bez zdalnej transmisji danych, z drugiej strony nie realizuje się również technik telekomunikacyjnych bez znaczącego udziału techniki komputerowej [2].

Rozwijająca się bardzo dynamicznie technologia mikroprocesorowa (co dwa lata ukazują się nowe generacje elementów, bardziej niezawodne w działaniu i coraz tańsze [8]), stwarza zupełnie nowe możliwości zarówno w stosunku do konstrukcji aktualnie eksploatowanych maszyn cyfrowych, jak również w zupełnie nowych obszarach działania. Przewiduje się np. zmianę funkcji wszelkiego rodzaju jednostek sterujących, stanowiących w dotychczasowych systemach komputerowych newralgiczne, z punktu widzenia niezawodności, węzły (np.: awaria jednostki sterującej pamięci dyskowej lub taśmowej, wyłącza z biegu od 6—8 jednostek tych pamięci). Funkcja nowych jednostek sterujących, jak przewidują trendy rozwojowe, zostanie ograniczona do translacji w zakresie złącz i kodów pomiędzy jednostką centralną i jednostkami pamięci. Natomiast inne funkcje, jak np. formatowanie informacji, ustalanie kolejek, kontrole parzystości itp. zostaną zwielokrotnione w poszczególnych jednostkach pamięci [6].

Tego rodzaju przesunięcie i zwielokrotnienie „inteligencji" w końcowych urządzeniach wpłynie na dalszą poprawę niezawodności, która i tak jest już obecnie oceniana bardzo wysoko.

Oficjalnie podawanym na Kongresie przez przedstawicieli różnych firm zachodnich (np. IBM) miernikiem niezawodności dla jednostek centralnych z 1 Mb PAO jest jedna awaria na rok, w czasie krótszym lub równym jednej godzinie (dla minikomputerów z PAO do 100 Kb powyższa wartość jest przyjmowana jeszcze o połowę mniejsza).

Z drugiej strony za efektywny okres eksploatacji (w korelacji z postępującym stale rozwojem technologii) jednej maszyny cyfrowej uważa się 5 lat [8].

Z technicznego punktu widzenia, nie ma już obecnie ograniczeń w zakresie wzbogacania dowolnej maszyny lub automatu w „inteligencję".

Ciekawym przykładem dotyczącym bardzo „wyrafinowanych" już form automatyzacji są tzw. samoobsługowe banki, które z technicznego punktu widzenia, zrealizowane są w oparciu o hierarchiczną sieć komputerową (trzystopniową) działającą bądź w trybie on-line (w godzinach otwarcia banków), bądź off-line (w godzinach zamknięcia banków). Terminal stanowią tzw. „bankomat" wyposażony w czytnik karty identyfikacyjnej, klawiaturę kodową, mały monitor ekranowy, małą drukarkę do pokwitowań oraz automat do wydawania i wpłacania pieniędzy. Terminal ten obsługiwany jest, bez jakiegokolwiek pośrednictwa osób trzecich, przez każdego klienta (osobę fizyczną), który wcześniej zgłosił do banku chęć korzystania z tego rodzaju usług.

W ramach tych usług, nie tylko jest realizowana funkcja wypłacania zadeklarowanej kwoty pieniędzy (do 5000 fr. szw. w trybie on-line lub 1 000 fr. szw. w trybie off-line), ale również wyprowadzanie informacji (na monitorze ekranowym lub papierze) o aktualnym stanie salda klienta, informacji o ostatniej wpłacie (dokonywanej np. poza klientem, przez zakład pracy, bezpośrednio na jego konto), potwierdzenie wpłaty pieniędzy, czy też np. zamawianie oficjalnego wyciągu bankowego lub druków bankowych, które w następstwie tego żądania dostarczane są do klienta pocztą. Mamy zatem do czynienia z zastąpieniem drogą automatyzacji funkcji normalnego stanowiska (okienka) bankowego z możliwością zapewnienia 24-godzinnej obsługi [9]. Istotnym elementem tej funkcji jest bezpośredni kontakt niewykwalifikowanej osoby fizycznej z komputerem, stanowiący jedną z głównych tendencji aktualnego rozwoju informatyki, co podkreślano w wielu referatach. Dla zrealizowania zadania samoobsługowego banku zaangażowano wiele

obecnie dostępnych środków technicznych informatyki (komputer IBM370 158-3 z PAO 4 Mb oraz pamięcią dyskową 3000 Mb, minikomputery IBM 3705 z PAO 90 Kb oraz pamięcią o bezpośrednim dostępie 0,5 Mb, jednostki sterujące IBM 3601 itd.) oraz teledacji. Ciekawie przy tym wygląda strona ekonomiczna zadania. Dla St. Gallische Kontonalbank (Szwajcaria) [9] eksploatującego obecnie 13 „bankomatów" rozmieszczonych przestrzennie w całym kantonie, system jest opłacalny już przy 1000 operacjach samoobsługowych miesięcznie.

Wyszczególniona wyżej sieć komputerowa nie tylko obsługuje „bankomaty", ale również wszystkie inne ważne obszary działalności banku, głównie w drodze zdalnego przetwarzania za pośrednictwem 170 różnych terminali.

W zakresie rejestracji danych stwierdzono, że najpowszechniejszym sposobem jest tu klawiatura bezpośrednio sprzężona z komputerem bądź zapewniająca zapis na standardowym nośniku magnetycznym [2].

Obecnie — w odróżnieniu do lat ubiegłych — w dalszym rozwoju techniki ujmowania danych nie przywiązuje się zbyt dużego znaczenia do stosowania czytników optycznych pisma OCR. Wiąże się to zapewne z faktem, że dla efektywnego i niezawodnego działania czytników OCR, trzeba wykorzystywać jako pośrednika maszynę do pisania (a więc znowu klawiaturę), a także jest to kwestia wysokich nadal kosztów urządzeń OCR. W każdym razie na Kongresie nie wygłoszono żadnego osobnego referatu, poświęconego tej technice wobec np. dwóch referatów poświęconych technice wyprowadzania danych na mikrofilm COM (Computer Output Microfilm), co jest bardzo znamienne.

Natomiast jako aktualnie dostrzegalny przełom technik wprowadzania (jak i wyprowadzania) informacji uważa się urządzenia foniczne. Przełom technologiczny w zakresie elementów elektronicznych, charakteryzujący się zarówno postępującą miniaturyzacją, jak i stałym trendem zmniejszenia się kosztów produkcji, znacznie przybliża zdaniem kilku referentów, rozwiązanie tego problemu [2]. Należy się przy tym liczyć z określonymi problemami, częściowo tylko natury technicznej. Wiąże się to z faktem, że w przypadku sterowania klawiatury oraz logicznej i zwięzłej symboliki mamy do czynienia z jasną, jednoznaczną oraz maksymalnie precyzyjną treścią informatyczną. Natomiast w przypadku „żywej mowy" niezależnie od uwagi oraz koncentracji osoby komunikującej się z komputerem, występują niejasności, niejednoznaczności oraz niedokładności, które w budowie detektorów analogowych oraz urządzeń towarzyszących rozwiązaniom fonicznym stworzą taki stopień komplikacji, że realizacja techniczna będzie zawsze bardzo kosztowna. Nie wyklucza to oczywiście możliwości zastosowania tych urządzeń, szczególnie tam, gdzie np. nie występuje niedobór kandydatów do obsługi klawiatury. Należy również zaznaczyć, że urządzenia komunikujące się bezpośrednio z komputerem za pomocą mowy, stworzą zupełnie nowe perspektywy techniczne (np. ogólnodostępna sieć telefoniczna służyć będzie do masowego wprowadzania i wyprowadzania informacji).

W zakresie aktualnie już znanych technik wyprowadzania informacji przewiduje się, że w stosunkowo niedługim czasie jakość wydruku komputerowego osiągnie jakość druku książki, przy czym również i w technice poligraficznej za ekstrawagancję uważać się będzie drukowanie tekstów bez wsparcia komputerowego [2].

Przejście od papieru do ekranu monitora stało się w wielu przypadkach faktem dokonanym, niemniej znikanie z ekranu obrazu tekstu, powoduje pojawienie się u użytkownika pewnej nieufności, w zestawieniu z realizmem zadrukowanego papieru.

Nie przewiduje się jednak, aby ekran mógł wyeliminować z użycia papier oraz inne techniki pochodne. Wszędzie tam, gdzie występuje potrzeba bardzo szybkiego dostępu do informacji, celem dokonania kilkusekundowej analizy jej treści, olbrzymią przewagę ma zastosowanie ekranowych monitorów. Natomiast wszędzie tam, gdzie informacja jest „stabilna" i potrzebna na dłuższy; okres, dominować będzie forma pisana, oczywiście z uwzględnieniem pośrednich technik, np. COM.

Ciekawe, że również i na Zachodzie dostrzega się dotkliwe skutki stert papieru obrastających każdy komputer, które mimo możliwości i dostępności stosowania innych technik emisji wyników, stanowią nadal niczym w gruncie rzeczy nie uzasadnioną rozrzutność. Mniejszą wagę przywiązuje się przy tym do samego problemu oszczędności papieru, a raczej widzi się pewien problem socjologiczny, polegający na wieloletnim oswojeniu i przyzwyczajeniu człowieka do pisanej informacji, chociaż z technicznego punktu widzenia można by już obecnie zrezygnować z wszelkich segregatorów pism i innej popierowej „otoczki" współczesnych biur.

W jednym z referatów [3] przedstawiono futurystyczny model biura dużej organizacji gospodarczej, w którym oprócz wszelkiego rodzaju wyposażenia informatycznego, zatrudnione będą jedynie dwie osoby, tzn. dyrektor oraz sekretarka, przy czym tej ostatniej przypisuje się jedynie funkcje nieprofesjonalne.

OPROGRAMOWANIE

Trendom rozwojowym oprogramowania poświęcono na Kongresie wiele uwagi. Zadecydowały o tym nie tylko takie względy, jak aktualnie kształtujące się koszty realizacji zamierzeń informatycznych, w których 75% stanowią koszty opracowania oprogramowania, ale również względy socjologiczne, przyjmowane powszechnie jako dostrzegalne już obecnie granice stosowania technik informatycznych [2, 6]. Niezawodność pracy sprzętu polegająca na niczym niezakłóconym, wielotygodniowym, bezawaryjnym funkcjonowaniu złożonych systemów komputerowych, pozwala na nową jakościowo i wartościowo ocenę efektywności działania człowieka, tj. programisty, operatora lub użytkownika systemu komputerowego, warunkujących przecież stopień wykorzystania sprzętu. Podano interesujący przykład ilustrujący zaobserwowany w ostatnich 10 latach wpływ technologii wytwarzania na rozwój przetwarzania informacji [6]:

- szybkość przepływu informacji w jednostkach centralnych komputerów (CPU) wzrosła 100 razy
- szybkość wykonywania operacji w CPU wzrosła 20 razy
- wskaźnik pojemności pamięci do czasu dostępu poprawił się 10 razy
- szybkość drukarek wzrosła 3 razy
- wydajność programisty wzrosła 1,2 razy.

Zastanawiające jest to, że wielu prelegentów przedstawiających praktyczne rozwiązania problemów informatycznych [4, 5], zapytanych o największe aktualnie trudności, jednoznacznie wskazywało małą efektywność pracy ludzkiej, w szczególności projektantów i programistów.

Mimo że ostatnie dziesięciolecie charakteryzowało się wprowadzeniem wielu nowych metod i technik programowania, takich jak programowanie narzędziowe i strukturalne oraz udoskonalenie języków wyższego rzędu, wzrost efektywności programowania, zwłaszcza w porównaniu z błyskawicznym rozwojem sprzętu, uważa się za wysoce niewystarczający. Czynione są daleko już zaawansowane wysiłki w kierunku zmniejszenia tej różnicy. Wskazane tu tendencje ujawniły się ostatnio w szczególnie drastycznym stopniu na tle prawdziwej rewolucji technologicznej w rozwoju elementów elektronicznych. Wskazano [6] np., że wobec aktualnie kształtującej się ceny opracowania i wytestowania jednej instrukcji programowej na poziomie 10 dol. USA oraz ceny jednego mikroprocesora (poniżej 10 dol. USA), zupełnie realne staje się zastąpienie w niektórych obszarach zastosowań tradycyjnych form oprogramowania — oprogramowaniem układowym. Przytaczane tu kierunki zastosowań dotyczą głównie oprogramowania narzędziowego (zarządzanie bazą danych, operacji WE/WY, dostęp do zbiorów, oprogramowanie procesorów komunikacyjnych).

Będzie to miało również duży wpływ na strukturę i funkcje przyszłych systemów operacyjnych, które będą coraz bardziej specjalizowane wg różnych typów zastosowań.

Przewiduje się również, że już w niedalekiej przyszłości pakiety programowe dla mini- i mikrokompute-

rów zostaną zrealizowane w postaci wymiennych układów elektronicznych. Coraz bliższa stanie się zatem wizja maszyn cyfrowych (przynajmniej dla wybranych zastosowań), które w ogóle nie będą wymagały programowania [2]. Ich oprogramowanie, „zaszyte" bądź na stałe, bądź w postaci wymiennych modułów, nie będzie wymagało zewnętrznej interwencji, zaś ich obsługa ograniczać się będzie do elementów mechanicznych (klawisze, pióra świetlne itp.) łatwych do opracowania nawet przez osoby niewykwalifikowane.

Ten nieco futurystyczny obraz [2] bierze swą podstawę w dobrze już ugruntowanym i udokumentowanym przeświadczeniu, że granice stosowalności komputera, określone są przez ludzi, którzy go skontrolowali, oprogramowali i aktualnie z niego korzystają. Ludzie zaś są istotami nie tylko popełniającymi błędy i pracującymi z określoną wydajnością, ale również posiadającymi wyobraźnię. Gdyby pracowali z absolutną perfekcją (do czego coraz bliżej zmierza współcześnie wytwarzany sprzęt), nie byliby również w stanie już nic nowego wymyśleć.

Poszukiwania kierunków poprawy efektywności programowania mają swoje praktyczne uzasadnienie. Uważa się za dobrą wydajność opracowanie i wytestowanie przez jednego programistę w ciągu jednego roku 7000 instrukcji programowych. Praktyka organizacji zajmujących się usługowo produkcją oprogramowania wskazuje [5], że ok. 68% wszystkich kosztów związanych z opracowaniem oprogramowania — to konserwacja systemów już istniejących.

Stąd wynikają również tendencje do zautomatyzowania wielu procedur programowych, przy przyjęciu nawet bardzo daleko idących ograniczeń w zakresie standaryzacji i pakietyzacji. W referacie [7] podano przykład możliwości budowy banku danych dla średnich i małych użytkowników, który poprzez zastosowanie niedawno zaanonsowanego pakietu dla komputera IBM system/38, pozwala „bezproblemowo" zbudować bank danych w stosunkowo krótkim czasie (ok. 1 roku) i to bez konieczności zmiany struktury danych wykorzystywanych w dotąd eksploatowanych systemach informatycznych. Oceniono, że w ten sposób zbudowany bank danych stwarza użytkownikowi ok. 80—90% wszystkich możliwości eksploatacyjnych, jakie mógłby on uzyskać w przypadku budowy od podstaw banku danych, ukierunkowanego na specyfikę jego zastosowań. Należy stwierdzić, że zbudowanie takiego banku trwałoby 3—4 lata i oprócz znacznie większych kosztów opracowania, zmuszałoby organizację wdrażającą do zatrudnienia specjalistów z zakresu budowy banku danych, co nie jest konieczne w przypadku zastosowania wspomnianego pakietu.

Jakie więc są szanse i granice przetwarzania informacji? Z przytoczonych relacji dotyczących trendów rozwojowych w zakresie zastosowań nowoczesnych elementów elektronicznych i sprzętu informatycznego wynika dosyć jednoznacznie, że jest nią postępująca automatyzacja, wkraczająca ostatnio również w dziedziny poza informatyczne.

Uważa się, że granice postępu w tym obszarze zastosowań informatyki leżą nie w ciągłym ulepszaniu i usprawnianiu konstrukcji sprzętu, ale raczej w sferze jej socjologicznego, a nawet politycznego oddziaływania. W wysoce już obecnie zinformatyzowanych państwach wyczuwa się pewną niechęć i obawy społeczeństwa przypisującego rozwojowi automatyzacji bezpośredni wpływ na zwiększające się stale bezrobocie, względnie możliwość naruszenia i ograniczenia wolności osobistej [1]. D. Hammer z USA (SERRY-UNIVAC) podał w trakcie dyskusji panelowej informacje, że obecnie w USA gromadzi się i przechowuje milion bitów informacji o każdym obywatelu tego kraju. Stąd zupełnie innego znaczenia nabrały problemy zabezpieczenia informacji, a w następstwie tego również aspekty prawne związane z manipulacjami, dostępem oraz wykorzystaniem informacji przez osoby niepowołane.

W zakresie rozwoju oprogramowania szanse widzi się głównie w poprawieniu efektywności programowania przez zwiększenie wydolności ludzkiej, jak i zastosowanie nowych metod oraz technik. Granice w tym obszarze oddziaływania wydają się być jeszcze wyraźniej określone, niż w przypadku sprzętu, przez czynnik ludzki.

Przytoczony już D. Hammer stwierdził, kończąc swą wypowiedź w dyskusji, że aktualny rozwój informatyki można porównać do jazdy samochodem po ulicy jednokierunkowej, z której nie można już zawrócić; nie widać również jej końca, lecz na każdym odcinku tej drogi obowiązywać winna pełna kontrola, o czym należy pamiętać już dzisiaj, a także w przyszłości.

LITERATURA

[1] Firnberg H.: Przemówienie z okazji uroczystego otwarcia VI Międzynarodowego Kongresu w Wiedniu, INFORMATYKA 7—8/1980

[2] Zemanek H.: Hoffnungen und Grenzen der Informations-verarbeitung

[3] Steinmueller W.: Informationstechnologien als soziales Problem und politische Herausforderung

[4] Kampe M.: Aufzeigen und Grenzen bei der Konzepterarbeitung des Distributing Processing an einem praktischen Beispiel in der chemischen Industrie

[5] Rupp W.: Erfolgskontrolle in einer EDV- Dieustleistungsunternehmung — Voraussetzung fuer wirtschaftliche EDV

[6] Vasko T.: Trends in the Development of Information Processing

[7] Schenk R.: Datenbanknutzen ohne Datenbankprobleme

[8] Stapleton P.: Plugcompatible Computers Trends, Opportunities and Chances

[9] Gabathuler W., Hurn H.: Chancen und Grenzen der Automation am Beispiel einer Selbstbedienungsbank

Teksty referatów od [2] do [7] zawarte są w opublikowanych materiałach Kongresu

Mgr inż. Józef PIASECKI
ZETO Olsztyn

Artykuły problemowe, recenzje, informacje, polemiki, wypowiedzi związane z nurtującymi Was, informatycznymi problemami...

Piszcie! Łamy INFORMATYKI otwarte dla wszystkich!

Kontrola w nowoczesnych systemach finansowo-księgowych

Hans-Thomas Schäfer pisząc swoją książkę „Revision bei Datenbanksystemen" [1]) postawił sobie za cel przedstawienie metod kontroli w systemach rozliczeń finansowo-księgowych realizowanych w oparciu o systemy automatycznego (komputerowego) przetwarzania danych, a w szczególności o systemy wykonane w technologii bazy danych. Swą uwagę skupił zwłaszcza na możliwości zastosowania pakietów oprogramowania zorientowanego problemowo do realizacji wymienionych funkcji kontrolnych.

Zdaniem autora konieczność podjęcia tego tematu wynika z faktu, że organy kontrolno-rewizyjne dużych systemów finansowo-księgowych, do których wprowadzono automatyczne przetwarzanie danych, zostały postawione w obliczu jakościowo nowych problemów. W nowej sytuacji dotychczas stosowane procedury kontrolno-rewizyjne okazały się niewystarczające albo zupełnie nieprzydatne.

W kolejnych częściach książki autor omówił następujące cztery zagadnienia:
— system automatycznego przetwarzania danych, stanowiący element systemu rozliczeń finansowo-księgowych, jako narzędzie wszechstronnej kontroli
— zadania organów kontrolno-rewizyjnych w warunkach zastosowania automatycznego przetwarzania danych
— zastosowania pakietów oprogramowania zorientowanego problemowo do realizacji funkcji kontrolnych w systemie finansowo-księgowym zrealizowanym w technologii bazy danych
— stan aktualny w zakresie wykorzystania dostępnych środków programowych do realizacji omawianych funkcji kontrolnych.

Książka adresowana jest zasadniczo do odbiorców rekrutujących się z dwóch różnych grup zawodowych, którzy spotykają się przy realizacji systemów rozliczeń finansowo-księgowych wspomaganych komputerem:

— projektantów systemów automatycznego przetwarzania danych z zakresu obliczeń finansowo-księgowych

— użytkowników tych systemów, a w szczególności organów kontrolno-rewizyjnych.

Książka buduje płaszczyznę porozumienia pomiędzy obydwoma grupami, proponując system wspólnych pojęć, pozwalających grupie przyszłych użytkowników na czynne włączenie się w proces projektowania systemu banku danych finansowo-księgowych.

Z punktu widzenia wymienionych grup odbiorców książka jest pozycją interesującą, którą można i należy polecić. Niewątpliwą zaletą książki jest aktualność i dobry poziom dyskutowanych rozwiązań z zakresu komputerowego przetwarzania danych. Autor poparł swoje rozważania bogatym wykazem literatury.

Książka wyróżnia się przejrzystą konstrukcją logiczną oraz dbałością autora o czytelność dzięki użyciu licznych tabel, zestawień i schematów ideowych. Szkoda, że w wielu miejscach niski poziom graficzny ilustracji obniża tę czytelność.

Maria WAŚKO

[1]) H.-Th. Schäfer: Revision bei Datenbanksystemen. STMV S. Toeche-Mittler Verlag, Darmstadt 1980, S. 230, DM 44.—

O jednolitą terminologię

„Programowanie"

Programowanie komputerów jest już dziedziną niemłodą i większość pojęć, jak również odpowiadających im nazw jest ugruntowana. Jednakże w wyniku nieustannego rozwoju środków programowania powstaje wiele nowych terminów jeszcze niezbyt dokładnie sprecyzowanych. Istnieją też terminy od dawna używane, którym jednak daleko — z punktu widzenia poprawności językowej — do postaci idealnej.

Ostatnio — na przykład — pojawił się termin j ę z y k w y s o k i e g o r z ę d u, z zamiarem wyparcia dotychczas używanego — j ę z y k a w y s o k i e g o p o z i o m u (ang. *high-level language*). Nie wydaje mi się, aby taka zamiana była uzasadniona. Pomijając celowość i skuteczność wypierania terminów powszechnie używanych przez inne, zatrzymam się nad poprawnością obu określeń.

Mówiąc ogólnie — omawiane pojęcie różnicuje języki programowania ze względu na ich przystosowanie do określonego typu komputera. Podstawowym kryterium zaliczenia języka, w powyższym sensie, do jakiejś kategorii, jest stopień jego zależności od budowy konkretnego komputera lub określonej klasy komputerów. Zatem języki najbardziej zależne od konkretnych komputerów, np. języki symboliczne (ang. *assembly languages*) cechuje niższy poziom (nie rząd) ogólności i (konsekwentnie) języki mniej zależne — wyższy poziom.

W językach bardziej zależnych od komputera pisze się oprogramowanie niższego poziomu (nie rzędu, bo nie ma rzędów oprogramowania), a w językach mniej zależnych — wyższego poziomu. Pojęcie rzędu języka mogłoby istnieć w praktyce programowania, gdyby zostało stworzone w teorii programowania i miało sens liczbowy tak, jak np. rząd równania w matematyce.

Określenie język wysokiego rzędu wydaje się niepoprawne także z przyczyn czysto językowych. O ile termin wysoki poziom nikogo nie razi, to wątpię w poprawność wysokiego rzędu. Jeżeli Amerykanie utworzyli grupę badawczą zajmującą się zagadnieniami utworzenia języka wysokiego poziomu i nazwali ją High Order Language Working Group, to nie znaczy, że ta nazwa jest poprawna.

Wyjaśnienia wymaga — w moim przekonaniu — także sprawa poprawnego posługiwania się terminem asembler. W języku polskim funkcjonuje on zarówno jako program tłumaczący (poprawnie), jak i język symboliczny (niepoprawnie). Aby rozwiać wątpliwości, posłużę się normą międzynarodową ISO 2382, która w sposób jednoznaczny precyzuje zakres obu pojęć. Asembler (ang. *assembly program*) jest to — według niej — program komputerowy wykorzystywany do zestawiania (w oryginale: *a computer program used to assemble*).

Język symboliczny (wg normy PN-71/T-01016 odpowiednik terminu *assembly language*) jest to język ukierunkowany maszynowo, którego rozkazy odpowiadają wzajemnie jednoznacznie rozkazom komputera i który umożliwia używanie makrorozkazów. Nazywanie języka symbolicznego asemblerem jest nieporozumieniem i wzięło się prawdopodobnie stąd, że niektórzy producenci określają język symbolicznie swoich komputerów mianem ASSEMBLER, lecz wówczas jest to tylko nazwa własna, a nie termin ogólny.

Jeżeli chodzi o używanie określenia język asemblera, to powinno być dopuszczalne, ponieważ jest choćby skrótową formą sformułowania: „język pisania programów tłumaczonych przez asembler". Natomiast za pełnoprawne (wobec terminu asembler) uważałbym określenie program tłumaczący, oczywiście w tych tylko przypadkach, w których nie zachodzi obawa nierozróżnienia różnych rodzajów programów tłumaczących.

Dalsze wątpliwości wzbudza termin asembler skrośny będący dosłownym przekładem angielskiego *cross-assembler*. Przypomnijmy, że jest to program tłumaczący wykonywany na komputerze innym od tego, dla którego tworzy kod wynikowy. Tej cechy programu chyba nie odzwierciedla przymiotnik „skrośny". Nie ujmując niczego precyzji określenia, wprost przeciwnie — zwiększając ją, proponowałbym wprowadzenie terminu asembler zewnętrzny, który jest bardziej poprawny językowo.

Polskie normy nie precyzują nazw wielu programów wchodzących w skład oprogramowania podstawowego (upowszechniający się polski odpowiednik angielskiego terminu *system software*). Często zdarza się, że na podstawie samej nazwy trudno zorientować się, o jaki program w istocie chodzi, brakuje bowiem jednoznaczności. Nie wiadomo — na przykład — jaki termin spośród następujących jest poprawny: dyrygent, egzekutor, monitor, program nadzorczy czy program zarządzający.

W innych wypadkach do języka polskiego przenoszone są żywcem terminy angielskie, np. *editor*, *loader*, itp., co jest niedopuszczalne. Od niedawna pojawiają się odpowiedniki polskie mające obcy źródłosłów, inny od określenia oryginalnego, co nie jest wyjściem zbyt szczęśliwym (np. koordynator — ang. *scheduler*, konsolidator — ang. *linker*).

Terminy program zarządzający i system operacyjny zostaną omówione oddzielnie. W każdym razie ustalając nazwę programu „sterującego wykonywaniem innych programów oraz przebiegiem pracy komputera" (ISO 2382) warto pamiętać o tym, że samo pojęcie nie jest jeszcze w pełni zdefiniowane, ponieważ nie ma jednolitej koncepcji co do funkcji tego programu.

W takich przypadkach, jak *editor* czy *debugger* istnieją odpowiedniki polskie: program redagujący (nie wydawniczy) i program uruchamiania. Należy tylko przestrzegać ich używania. Podobnie ma się rzecz z terminem *loader* (program ładujący, a nie ładowacz). Gorzej z terminem *bootstrap*, który może w ogóle nie doczekać się polskiego odpowiednika. Jeżeli jednak byłaby taka potrzeba, to można zaproponować określenie program samoładujący, ponieważ bootstrap spełnia funkcje programu ładującego w odniesieniu do samego siebie.

Przez wiele lat odpowiednikiem angielskiego terminu *linker* był program łączący i nie widzę powodu, dla którego należałoby tę nazwę zmieniać na konsolidator, użyty w kilku niedawno wydanych publikacjach książkowych. Pomijając spór, czy konsolidacja i łączenie jest tym samym, należy stwierdzić, że zastępowanie wyrazów angielskich polskimi odpowiednikami o obcym źródłosłowie jest praktyką niebezpieczną. Chciałbym widzieć minę tego, kto wymyślił termin konsolidator, gdyby w terminologii angielskiej pojawiło się określenie *consolidator*. Podobnie ma się sprawa z wyrazem koordynator, jako odpowiednikiem angielskiego terminu *scheduler* (program szeregujący).

Ten ostatni termin *(scheduler)* jest na tyle ważny i rzadko precyzyjnie określany, że warto podać jak się go rozumie w normie ISO. Będzie to o tyle utrudnione, że przedtem należy znaleźć odpowiednik terminu *dispatcher*, przy użyciu którego zdefiniowano program szeregujący. *Dispatch* znaczy dosłownie wysyłka lub spedycja. Natomiast norma ISO 2382 podaje następujące określenie: *to dispatch* — przydzielać czas procesora pracom lub zadaniom gotowym do wykonania. Z kolejnego określenia wynika, że *dispatcher* jest programem, którego zadaniem jest przydzielanie procesora w sensie określonym w poprzednim zdaniu. Mówiąc inaczej *dispatcher* jest programem, który wykonuje funkcję związaną z przekazaniem sterowania nowemu programowi. W literaturze polskiej znalazłem kilka prób zastąpienia terminu *dispatching* (nadzorowanie, przydzielanie, uruchamianie), jednak żadna z tych nazw nie jest jednoznaczna.

Ze względu na fakt, że *dispatcher* zajmuje się ekspediowaniem, wysyłaniem programów (zadań, procesów) do procesora (w ogólniejszym przypadku — do procesorów), a więc jakby ich rozprowadzaniem, właśnie tak określiłbym jego funkcję, a sam program nazwałbym programem rozprowadzającym. Odpowiada to treści pojęcia i jest zgodne np. z określeniami francuskimi z tej samej normy: *répartir*, *distribuer*. Według niej czasownik to *schedule* znaczy „wybierać prace lub zadania, które będą rozprowadzane". Takiemu określeniu czasownik szeregować odpowiada dokładniej niż mający szersze znaczenie — koordynować. Prostą konsekwencją tego stwierdzenia jest nazwanie programu wykonującego wspomnianą funkcję programem szeregującym.

Nazywanie programów według zasady program + imiesłów czynny lub bierny (lub program + przydawka dopełniaczowa) jest przyjęte w języku polskim. Wystarczy zajrzeć do normy PN-71/T-01016 lub którejś z ostatnio wydanych książek (np. A. C. Shaw, Projektowanie logiczne systemów operacyjnych; WNT, Warszawa, 1980). Choć takimi terminami, jak: translator, kompilator, asembler lub symulator na pewno łatwiej się posługiwać, to nie widzę konieczności określania każdego programu pojedynczym wyrazem. Nie mamy w rodzimym języku mechanizmów ku temu, więc nie ulegajmy manierze.

Terminologia dotycząca sfery oprogramowania jest tyleż ważna, co kontrowersyjna, jako że nie ma informatyka, który by nie programował. Istnieje — oczywiście — wiele innych terminów wartych omówienia lecz o nich — przy innej okazji.

Janusz ZALEWSKI

Bibliografia wydawnictw polskich z dziedziny informatyki

Mikroprocesory — RZYMKOWSKI K., Wyd. ROINTE Energetyki i Energii Atomowej, Warszawa 1979, s. 223

Wprowadzenie do minikomputerowej techniki przetwarzania informacji. Układy dużej skali integracji (LSI). Organizacja logiczna wybranych typów mikrokomputerów. Mikroprocesory bipolarne. Wybrane zagadnienia związane z eksploatacją układów LSI MOS. Materiały przeznaczone są dla projektantów i użytkowników mikroprocesorów.

System MERA 9150, opis ideowy — Wyd. Warszawskich Zakładów Urządzeń Informatyki MERAMAT, Warszawa 1979, s. 66

Monitor ekranowy. Moduły operacyjne systemu MERA 9150. System przygotowania danych. Listy HELP. Sterowanie procesem przygotowania danych. Modus super-operatora. VALIDATOR — język programowania systemu MERA 9150. Wprowadzenie danych. Materiały przeznaczone są dla użytkowników systemu MERA 9150.

MERA 9150. Materiał metodyczny. Organizacja ośrodków przygotowania danych — Wyd. Warszawskich Zakładów Urządzeń Informatyki MERAMAT Warszawa 1979, s. 20

Taśmy systemowe MERA 9150. Zarządzanie przepływem danych. Obowiązkowe procedury codzienne. Dodatek: Dokumentacja kontrolna użytkownika.
Materiały przeznaczone są dla użytkowników systemu MERA 9150.

Z prac nad zastosowaniem elektronicznej techniki obliczeniowej w przedsiębiorstwach obrotu środkami produkcji. Komentarz do Uchwały nr 47/73 Rady Ministrów — GULA M., Wyd. Państwowych Magazynów Usługowych, Warszawa 1979, s. 88, cena 29 zł Seria III: Jak programować efektywne wykorzystanie zasobów surowcowych i materiałowych

Charakterystyka elektronicznego systemu przetwarzania danych. Przyjęta metodyka projektowania i wdrażania typowego Zautomatyzowanego Systemu Zarządzania (ZSZ) o budowie modułowej. Typowy, zautomatyzowany system zarządzania przedsiębiorstwem hurtowego handlu środkami produkcji. Środki techniczne eksploatacji systemu. Przygotowanie przedsiębiorstwa obrotu środkami produkcji do wdrożenia systemu. Nakłady finansowe i efekty ekonomiczno-organizacyjne. Rozpowszechnienie systemu. Bibliografia. Słowniczek terminologiczny.
Publikacja o charakterze popularnym przeznaczona jest szczególnie dla dyrektorów, głównych księgowych, kierowników działów i branżystów central handlowych i wchodzących w ich skład przedsiębiorstw hurtowego handlu zaopatrzeniowego.

Jednolity System Elektronicznych Maszyn Cyfrowych — oprogramowanie OS/JS, SPM/JS. System programowania matematycznego. Podręcznik programisty systemowego, Wyd. Instytutu Komputerowych Systemów Automatyki i Pomiarów, Wrocław 1979, s. 10.

Budowa taśmy dystrybucyjnej systemu SPM/JS. Części SPM, MARVEL, SPMRG.

Jednolity System Elektronicznych Maszyn Cyfrowych — oprogramowanie OS/IS, GPSS/JS. System symulacji ogólnego zastosowania. Podręcznik programisty systemowego, Wyd. Instytutu Komputerowych Systemów Automatyki i Pomiarów, Wrocław 1979, s. 33.

Struktura, włączanie i eksploatacja systemu GPSS/JS. Dodatki: Wyprowadzenie na karty programu przykładowego. Komunikaty o błędach. Uwagi dotyczące włączenia bloku HELP.

Podstawy systemów operacyjnych — HANSEN P. B., tłum. wyd. ang. z 1973 r., WNT, Warszawa 1979, s. 351, cena 85 zł

Przegląd systemów operacyjnych. Procesy sekwencyjne. Procesy współbieżne. Zarządzanie procesorami i pamięcią. Algorytmy szeregowania. Ochrona zasobów. Analiza przykładu: RC 4000.
Książka przeznaczona jest dla projektantów systemów operacyjnych — programistów — oraz studentów wydziałów informatycznych wyższych uczelni.

Maszyny cyfrowe. Elementy i podstawy organizacji — KOSEK E., RESZKA G. S., PWN, Warszawa 1979, s. 272, cena 28 zł

Podstawy arytmetyczne. Elementy i układy logiczne EMC. Bloki funkcjonalne maszyny cyfrowej. Organizacja maszyny cyfrowej i jej pracy.
Książka przeznaczona jest dla osób stykających się po raz pierwszy z problemami konstrukcji i organizacji elektronicznych maszyn cyfrowych. Przedstawia w sposób przystępny podstawowe problemy maszyn cyfrowych.

Modernizacja systemu informacyjnego w przedsiębiorstwie przemysłowym — NOWICKI A., PWE, Warszawa 1979, s. 234, cena 42 zł Seria: Informatyka w praktyce

Wybrane problemy systemu informacyjnego. Programowanie prac modernizacyjnych. Identyfikacja i analiza systemu informacyjnego. Realizacja procesu modernizacji. Dodatek: Automatyzacja analizy. Ze względu na zdecydowanie metodyczny charakter, książka jest w głównej mierze przeznaczona jako pomoc dla studentów wyższych uczelni, na których jest wykładana problematyka projektowania systemów informatycznych. Ponadto jest adresowana do kierownictwa oraz służb informatycznych obiektów gospodarczych, w których są — lub będą — usprawniane funkcjonujące systemy informacyjne dla potrzeb nowoczesnego zarządzania.

Programy analizy konstrukcji inżynierskich za pomocą maszyny — GIER, KAMIŃSKI R. i inni, Wyd. Politechniki Warszawskiej, Warszawa 1979, s. 489. Prace nad automatyzacją projektowania konstrukcji inżynierskich (4)

Obliczanie parametrów wytrzymałościowych przekrojów betonowych belek zginanych (YB — 1AL). Obliczanie rzędnych linii wpływu sił wewnętrznych w belkach ciągłych (YB-2AL). Analiza sił wewnętrznych i przemieszczeń układów prętowych płaskich. Analiza statyczna rusztów przegubowych (RK-39), kratownic płaskich i przestrzennych (RK-47), ramownicy przestrzennej (RK-40). Materiały przeznaczone są dla inżynierów projektantów w dziedzinie budownictwa lądowego.

Urządzenia i systemy sterowania cyfrowego — BOGUSZ Z., Wyd. Politechniki Gdańskiej, Gdańsk 1979, s. 150, cena 15 zł

Wiadomości wstępne. Organizacja systemów sterowania cyfrowego. Układy wejść i wyjść wielkości ciągłych. Modułowy system CAMAC. Systemy automatyzacji stanowisk hamowniczych silników spalinowych. Skrypt przeznaczony jest dla studentów IV roku elektroniki, o specjalności automatyka.

Systemy liczące — SZKLANNY J., Wyd. Politechniki Gdańskiej, Gdańsk 1979, s. 129, cena 14 zł

Przetwarzanie zadań w systemach liczących. Algorytmy wykorzystania czasu pracy procesora w przypadku zadań niezależnych. Problemy wykorzystania pamięci w systemach liczących. Opis systemów liczących. Systemy wieloprocesorowe.
Skrypt przeznaczony jest dla studentów III i IV roku informatyki.

CSL — Język do symulacji zdarzeń — WAJS W., Wyd. Akademii Górniczo-Hutniczej im. S. Staszica w Krakowie, Kraków 1979, Wyd. 2 uzup., s. 203, cena 12 zł, skrypty uczelniane nr 696

Podstawowe elementy języka CSL. Pojęcie zbioru i definicja zbiorów w języku CSL. Wyrażenia arytmetyczne. Instrukcje. Generowanie liczb pseudosylabowych za pomocą generatorów. Instrukcje we/wy. Bloki działań w programie. Modelowa realizacja biegu czasu. Kompilacja programów i przykłady programowania w języku CSL. Uruchamianie programów symulacji w języku FORTRAN.
Skrypt przeznaczony jest dla studentów Wydziału Elektroniki, Automatyki i Elektroniki AGH, jako pomoc do wykładów z przedmiotów dotyczących modelowania procesorów oraz języków symulacyjnych.

Gdzie jest informatyka?

To pytanie zadają sobie dzisiaj zapewne nie tylko informatycy. Gdzie jest informatyka w życiu społecznym naszego kraju? Jaki jest jej udział w polskiej gospodarce? Buńczuczne obietnice i zapewnienia składane do niedawna decydentom przez pełnych optymizmu fachowców przychodzi teraz — już bez zawoalowań — skonfrontować z realiami. Niejeden ze znawców zakrzyknął, że „król jest nagi" — kolejna polaryzacja emocji. Chwila zaś jest szczególna; przygotowywane są właśnie założenia Reformy Gospodarczej.

Czy informatycy znajdą swoje miejsce w prowadzonych obecnie pracach, czy informatyka zostanie w Reformie uwzględniona? — to drugie pytanie (obok postawionego w tytule), na które odpowiedzi miało udzielić spotkanie czołówki polskich informatyków, zorganizowane 30 października br. przez Sekretariat Komitetu Informatyki. W dyskusji, której główne tezy przedstawione zostały poniżej, uczestniczyli: **Andrzej Dąbkowski, Konrad Fijałkowski, Bolesław Gliksman, Marek Greniewski, Janusz Gwiazda, Andrzej Janicki, Jerzy Kisielnicki, Juliusz Kulikowski, Jan Macieja, Jan Siery, Zbigniew Substyk, Zenon Szoda, Jerzy Trybulski, Władysław Turski, Stanisław Waligórski** oraz **Jan Żydowo.** Znaleźli się w tym gronie przedstawiciele przemysłu komputerowego, ośrodków naukowych, instytucji świadczących usługi informatyczne oraz bezpośrednich użytkowników informatyki. Ich poglądy wydają się reprezentatywne dla całego środowiska.

Załamanie gospodarki, jakie obecnie przeżywamy, stawia informatyków w trudnej sytuacji. Wykorzystywana przez nich kosztowna technologia nie dała — jak dotąd — spodziewanych efektów. U zainteresowanych dominuje więc raczej uczucie zawodu. Z drugiej zaś strony niebagatelny jednak potencjał informatyczny może być racjonalnie wykorzystany, bez wysokich dodatkowych nakładów inwestycyjnych. Niemniej, trzeba ujawnić wszystkie deformacje dotychczasowego rozwoju informatyki, co być może pozwoli określić podstawy przyszłych zadań. Znając swoje rzeczywiste możliwości (tj. budując plany na faktach, nie iluzjach), informatycy mogliby śmielej oferować społeczeństwu swoje usługi.

Informatyka jest — biorąc pod uwagę dzisiejszą sytuację — raczej dziedziną przyszłości, być może nawet przyszłości dość odległej. Opóźnienie w stosunku do państw wysoko rozwiniętych jest w tym zakresie coraz większe i nie sposób przewidzieć, kiedy znajdzie się ona w Polsce w powszechnym użyciu. Na pewno jednak nie nastąpi to prędko. Ponadto informatyka wymaga precyzyjnej organizacji informatyzowanych dziedzin oraz dostarczania prawdziwych danych. W ciągu całej historii elektronicznej techniki obliczeniowej w Polsce warunki te nie były spełnione. Informatycy wprawdzie — powołując się na cywilizacyjną rolę swej technologii oraz ogromne jej osiągnięcia w krajach rozwiniętych gospodarczo — zdołali przekonać decydentów o swojej opatrznościowej roli, nie wzięli jednak pod uwagę irracjonalnych mechanizmów polskiej nauki i gospodarki i — przegrali.

J. Kulikowski stwierdził, że podstawowym błędem informatyki było utrzymywanie zjawiska dysproporcji pomiędzy jej rozwojem a sytuacją społeczno-ekonomiczną kraju. Zawsze bowiem brakowało informatykom umiejętności przewidzenia tej sytuacji na 3—5 lat wcześniej, co pozwoliłoby na odpowiednie zagospodarowanie środków. Szczególnie charakterystyczny był rozdźwięk pomiędzy wyśrubowanym programem informatyki z początku lat siedemdziesiątych a możliwością realizacji. Połowa tej dekady dobitnie wykazała utopijny charakter podobnych zamierzeń. Ważny jest też proces jaki wystąpił potem. Otóż próbowano obronić niektóre elementy starego programu. Na czoło wysunęło się w rezultacie kilka systemów rządowych, akurat wtedy, gdy gospodarka narodowa zaczęła szukać dróg decentralizacji. I dzisiaj, przy załamaniu gospodarki, funkcjonujące systemy są w zasadzie bezużyteczne... *„CENPLAN nie jest w stanie podołać zadaniom bilansowania gospodarki, PESEL-MAGISTER nie jest w stanie zapobiec bezrobociu wśród kadry z wyższym wykształceniem, SPIS został generalnie skrytykowany za nierzetelne informacje o poziomie życia kraju itd.".*

Kilka osób wskazało na inny aspekt tej sytuacji. Skonstatowano bowiem, iż w rzeczywistości wartość informacji — w sensie ekonomicznym — była do tej pory prawie żadna. W. Turski: *„Sposób zarządzania naszą gospodarką jest niezwykle nieczuły na informacje".* M. Greniewski: *„Moim zdaniem, całe zarządzanie odbywa się praktycznie bez informacji. Dam przykład. Istnieje olbrzymia sprawozdawczość związana z dostawami na rynek wewnętrzny — zajmuje się nią ok. 10 tys. ludzi. Informacja końcowa ma postać agregatów w układzie resortowym — nie jest powiedziane jakie przedsiębiorstwo nie wykonało dostaw, a tylko — w jakich grupach towarowych ich nie wykonano. Co z takich informacji wynika? Jakie decyzje można na podstawie tego podjąć? Poza tym informacja ta dociera na szczebel rządowy w 27 dni po zakończeniu kwartału, czyli w terminie, w którym wypłaca się już pracownikom premie".* W sytuacji, gdy brakuje informacyjnego mechanizmu porządkującego sprawę dostaw, na informatykę nie ma w ogóle zapotrzebowania.

Na tle systemu gospodarczego, który w większym stopniu określany był na woluntarystycznych „ideologicznych" przesłankach, niż na podstawach rachunku ekonomicznego, matematyce i zdrowym rozsądku, informatyka nie mogła zabłysnąć, choć kreowano ją czasami na primabalerinę. Nie zabłysnęła zresztą nie tylko ze względu na warunki zewnętrzne. J. Kulikowski: *„Powstało bardzo silne lobby przemysłowe, bo przemysł jest w stanie zadecydować o wielu sprawach i to wbrew interesom użytkowników informatyki. Drugie lobby reprezentowane jest przez jednostki dyspozycyjne rządowych systemów informatycznych (głównie RCI PESEL), które nigdy, w żadnej formie, nie pozwoliło wejrzeć w swoje wewnętrzne sprawy".*

Koteryjne podziały niosą duże niebezpieczeństwo, gdy pozwalają ukryć jakieś istotne niedomagania, w tym przypadku — systemów informatycznych, pochłaniających duży procent wszystkich środków technicznych. Systemy rządowe są już teraz pod silnym ogniem krytyki. M. in. B. Gliksman stwierdził, że PESEL jest systemem bezużytecznym, gdyż ze względu na ograniczone możliwości technicz-

ne nigdy nie będzie w stanie prowadzić ewidencji wszystkich obywateli; już choćby samochody są zapisane u nas tylko w 30—35%. Podobnie „nieżywe" są — według B. Gliksmana — SPIS oraz podsystem MAGISTER.

Niejako w konkluzji B. Gliksman dodał, że polscy programiści lepiej sprawdzają się w tworzeniu systemów ewidencyjnych i że tylko w tym zakresie dysponujemy niezbędną siłą. Wynikałoby stąd, że wielkie systemy sterowania optymalizujące zarządzanie są poza naszymi możliwościami. *„Pakiety programowe były i są nienowoczesne, bez budowy strukturalnej. W nielicznych tylko przypadkach nadają się do rozpowszechniania. Nasze systemy budowane są prymitywnie. Działacze centralni chętnie podkreślają, że mamy świetną kadrę programistów, nie ustępującą programistom na Zachodzie. To nieprawda. Programiści nie stosują języków programowania wyższych rzędów, uwalniających ich od konieczności szczegółowego opisu procedur..."*. Z wielu wydruków użytkownicy nie korzystają w ogóle, gdyż systemy zostały po prostu źle zrobione. B. Gliksman dał przykład własnego przedsiębiorstwa.

Obraz sytuacji, czy raczej tylko jej szkic, dopełnia zmniejszanie się zasobów sprzętu komputerowego oraz duża zawodność sprzętu produkcji krajowej, stanowiącego podstawę polskiej informatyki. Istotnym dopełnieniem jest również fakt rezygnacji wielu przedsiębiorstw z eksploatowania dotychczas wykorzystywanych systemów. Mówił o tym Z. Substyk. Dzieje się tak w wyniku presji obecnych wydarzeń społecznych w kraju. Drastyczne ograniczenie wydatków inwestycyjnych i administracyjnych, zmiany w charakterze produkcji wielu zakładów, naciski robotników w fabrykach powodują, że dyrekcje są zmuszone do wycofywania zamówień na usługi informatyczne.

Informatycy muszą zatem odpowiedzieć na pytania zasadnicze. Co w ogóle robić? Jak opowiedzieć się wobec przygotowywanej Reformy Gospodarczej? Z jakimi propozycjami wystąpić? Uczestnicy dyskusji byli jednomyślni w tym, że środowisko nie może pozostać bierne wobec zachodzących zmian.

J. Kulikowski: *„Wydaje mi się, że słuszne jest, ażeby informatycy byli informowani o tym, jak rozwijają się prace komisji rządowej, która przygotowuje projekt reform gospodarczych i żeby w odpowiednim momencie można było go rozszerzyć czy uzupełnić w kierunku informatyzacji, ale skrojonej w takim zakresie, w jakim to będzie realne i możliwe"*.

W. Turski: *„Informatycy powinni wziąć udział w dyskusji ogólnonarodowej o tym, jak zreformować system zarządzania i przypuszczam, że mogą do niej wnieść istotny wkład. Natomiast bardzo bym przestrzegał przed robieniem reformy gospodarczej w taki sposób, aby była wygodna dla informatyki. To świat naszej gospodarki powinien być na tyle rozsądny, żeby w nim informatyka naturalnie wyglądała"*.

Z. Szoda: *„Całkowicie zgadzam się z tym, że nie można systemu zarządzania dostosowywać do informatycznych wyobrażeń o gospodarce, tylko odwrotnie. Po prostu trzeba z informatyką wejść do rzeczywistości, jaka by ona nie była"*.

J. Macieja namawiał z kolei, aby nie czekać na rozstrzygnięcia w sprawie reformy, lecz dołączyć się do prowadzonych prac. Podał przy tym przykład organizacji systemu zarządzania na szczeblu centralnym, który podzielony na drobne komórki organizacyjne, nie daje się już właściwie zinformatyzować; komputer nie wnosi tu wartości merytorycznych. Już pierwotna struktura organizacyjna powinna bowiem uwzględniać możliwości informatycznej technologii. *„Później jest za późno"*.

●

Konieczność oparcia zastosowań informatycznych na ścisłym rachunku ekonomicznym stała się już dla wszystkich oczywista. Wypada się już chyba pogodzić z faktem, że informatyka światowa wyprzedzi nas o dalsze lata, że być może zapóźnień nie sposób będzie odrobić. Nie można informatyką wymuszać postępu, musi zachodzić — twierdzi się to teraz powszechnie — proces odwrotny. Dlatego też trzeba uważnie śledzić zmiany w gospodarce i — przewidując jej rozwój — przygotowywać własne środki.

Funkcjonowanie rynku wyeliminuje zapewne wiele anomalii, charakterystycznych dla polskich zastosowań informatyki. Zmiany, które nastąpią, można z grubsza przewidzieć. Informatyka w zarządzaniu i sterowaniu przemysłem przestanie zapewne odgrywać główną rolę, być może w centrum zainteresowania znajdą się zastosowania powszechne, służące bezpośrednio społeczeństwu.

J. Trybulski stwierdził, iż w warunkach zarządzania autokratycznego branża mogła się rozwijać, jeśli przekonała decydentów do swych poczynań — stąd przewaga zastosowań przeznaczonych dla zarządzania (wbrew potrzebie i możliwościom). *„Brakuje natomiast masowych zastosowań. Moim zdaniem — nadszedł czas dla zastosowań banalnych, masowych i wysoce efektywnych. Na dworcach, pocztach, w sklepach (...) Nadszedł taki czas, że zastępowanie pracy żywej wysoce wydajną pracą komputera zaczyna mieć sens"*.

Taki kierunek rozwoju jest tym bardziej racjonalny, że wymagający stosunkowo niewielkiego potencjału środków technicznych. W tak trudnym gospodarczo okresie jest to chyba wyjście jedyne. Ponadto trzeba rozwijać zastosowania w tych wąskich dziedzinach, w których informatyka już się przyjęła i która jest w nich nie do zastąpienia. Wskazano dziedziny dzisiaj priorytetowe — W. Turski: astronomia, fizyka, wojsko, ruch lotniczy; B. Gliksman: operacje bankowe; M. Greniewski: rozliczanie rent i emerytur, państwowa sprawozdawczość statystyczna, sprawozdawczość bankowa, analiza dostaw na rynek podstawowych (1500—2000) towarów; K. Fijałkowski: informacja naukowa i techniczna.

W sytuacji ostrego „zaciśnięcia pasa" wielkie inwestycje informatyczne są oczywiście nierealne, niemniej możliwe jest efektywniejsze wykorzystanie obecnych środków. J. Kulikowski zwrócił w tym względzie uwagę na potrzebę uporządkowania prawno-organizacyjnej strony informatyki, co związane jest z precyzyjną koordynacją działań tej dziedziny. W ramach takiej koordynacji powinny się znaleźć systemy rządowe, do tej pory — jak już wspomniano — nie podlegające zewnętrznej ocenie. Konieczne jest także opracowanie jednoznacznych normatywów w informatyce (zatrudnienia, wykorzystania sprzętu, metodyk projektowania). Z. Substyk wskazał dodatkowo konieczność przeprowadzenia inwentaryzacji istniejącego sprzętu (głównie ok. 500 instalacji maszyn ODRA) i opracowania programu rozsądnego ich doposażania. B. Gliksman przypomniał natomiast o tym, że niedobór środków eksploatacyjnych paraliżuje już dzisiaj pracę ośrodków obliczeniowych, a pogłębienie tego zjawiska może zakończyć wszystkie rozważania o przyszłości informatyki.

Uporządkowanie spraw informatycznych (oprogramowanie, sprzęt, kadra) musi nastąpić jak najszybciej, na innych niż dotąd zasadach. J. Kulikowski: *„Nie wierzę w*

to, żeby dzisiaj można było wrócić do pełnego centralizmu koordynacji rozwoju informatyki, w moim przekonaniu byłoby to akurat sprzeczne z wszelkimi tendencjami, które obserwujemy w gospodarce narodowej. Wprawdzie koncepcja, aby był jeden centralny organ koordynujący na tak wysokim szczeblu, by mógł wpływać na decyzje ministrów, nie była zła, niemniej jej realizacja była błędna. Podporządkowanie tego organu premierowi oznaczało, że będzie tym rządził ktoś, kto ma zbyt wiele spraw na głowie. Dlatego też powinno to być przekazane jednemu z wicepremierów, który odpowiadając za węższy krąg zagadnień, mógłby bardziej aktywnie kierować sprawami informatyki. Jeśliby z kolei sprowadzić organ koordynujący na poziom pojedynczego resortu, jak to ma miejsce obecnie, to po prostu nie będzie on w stanie spełniać swoich zadań. Informatyce jest jednak potrzebna koordynacja centralna. I tam muszą zapaść decyzje o tym, które maszyny produkować, a których nie, co importować, w jakim zakresie uczestniczyć w kooperacji międzynarodowej, w jakim wymiarze produkować za pomocą własnych środków.

(...) W tej chwili — sądzę — musimy wystąpić z jakimś programem, który będzie obejmował całość zagadnienia (tzn. nie będzie mówił tylko o zastosowaniach, a pomijał sprawy produkcji sprzętu, itp.). Musi to być program bardzo skromny, stwarzający pewne kierunki preferencji. Można by tu nie zaczynać od zera, lecz wrócić do dokumentów, które były już wcześniej opracowane".

●

Rozważając kwestie produkcji sprzętu komputerowego uczestnicy wyraźnie zgodzili się w jednym punkcie: trzeba precyzyjnie określić sfery produkcji, w której mamy się specjalizować. M. Greniewski wskazał pięć takich elementów: procesory komunikacyjne, modemy (choćby tylko wolne), terminale (głównie monitory ekranowe), system MERA 9150 wraz z drukarką wierszową DW-3 oraz pamięci dyskowe. Są to w większości produkty, których nie da się (MERA 9150) lub nie opłaca (monitory) eksportować. Powinny one przede wszystkim zaspokajać olbrzymie potrzeby krajowe. Uporządkowanie tych pięciu spraw, a także rozwiązanie problemu materiałów eksploatacyjnych daje — według M. Greniewskiego — szansę, by informatyka mogła u nas w przyszłości spełniać choćby najprostsze oczekiwania społeczne.

J. Żydowo ostro zaprotestował przeciw decyzji (*„nawet nie wiadomo, kto te idiotyczne decyzje podejmuje"*) o zaprzestaniu budowy komputerów ODRA. Zwiększenie zakresu wyposażenia tych maszyn o 25% ich wartości dałoby 3—4-krotny wzrost mocy obliczeniowej. Poza tym dokonanie wymuszonej konwersji systemów z 500 instalacji odrowskich na maszyny Jednolitego Systemu będzie łącznie kosztowało powyżej 4 mld zł, czyli prawie tyle, ile wynosi aktualna wartość (po uwzględnieniu amortyzacji) zainstalowanych w kraju zestawów ODRA. *„Przemysł powinien produkować nie to, co chce, ale to, co wynika z potrzeb użytkownika. Zrobiono odpowiednie ankiety, które są kompletnie lekceważone, nikt się tymi potrzebami nie zajmuje. Tylko się produkuje, ale nie wiadomo w zasadzie dlaczego (...) Nie wolno zatrzymywać produkcji ODRY, trzeba produkować ją tak długo, jak to będzie potrzebne".*

J. Macieja: *„Stopa zysku przedsiębiorstw w Zjednoczeniu MERA jest tak wysoka, że ustępuje jedynie stopie zysku produkcji alkoholu. Wynosi ona 55%!"* A. Janicki: *„Czy nasz przemysł komputerowy cieszy się z tego? Nie. Było to nam narzucone. Musimy w ten sposób dopłacać do innych przemysłów. Stąd nasze wyroby są jeszcze stosunkowo drogie".*

W dyskusji bardzo wyraźnie podkreślono, że główną podstawą do wyprowadzenia kraju z olbrzymiego zadłużenia i gospodarczej recesji jest przemysł maszynowy. Przemysł ten jednak niewiele zdziała, jeśli nie zostanie zasilony mikroprocesorami (automatyką sterowania). J. Macieja: *„Zgadzam się z tymi, którzy mówią, że łatwe czasy się skończyły i za parę lat niczego, co się rusza: koparki, samochodu, obrabiarki, maszyny do szycia — nie będzie można sprzedać (poza kilkoma krajami bardzo słabo rozwiniętymi), jeśli nie będzie miało wmontowanych urządzeń sterowania mikroprocesorowego".*

A. Janicki: *„My jako kraj średniej wielkości najprawdopodobniej nigdy nie będziemy wielkim producentem szerokiego asortymentu mikroukładów wielkiej i bardzo wielkiej skali integracji. Nasza gospodarka bowiem nie wytrzymałaby tego ekonomicznie. Niemniej krajowy przemysł elektroniczny musi być liczącym się producentem wybranej gamy mikroukładów elektronicznych i to o najwyższym współczynniku nowoczesności. Dotyczy to zwłaszcza nawet małoseryjnej produkcji wysoce wyspecjalizowanych układów hybrydowych, wielomikrostrukturowych i monolitycznych, niezależnie od jednostkowego kosztu wytwarzania. Mogłoby się to wydawać nieekonomiczne, ale posiadanie możliwości wbudowania takich mikroukładów do urządzeń powszechnego użytku istotnie zmienia tę kalkulację".*

W podsumowaniu tej kwestii W. Turski: *„Z produkcji mikroprocesora nic nie wyjdzie, dlatego że mikroprocesory co rok zmieniają swój wystrój, swoje walory użytkowe — to jest kosztowny czubek piramidy. Próbowanie budowy samego czubka piramidy bez jej podstawy — bez mechaniki precyzyjnej i elektroniki — zawsze będzie kosztowne. Oprócz tego — ze względu na konieczność kopiowania — produkt będzie niekonkurencyjny. Środki na tworzenie tego przemysłu znacznie przewyższają środki niezbędne na zakup mikroprocesorów, które będą potrzebne dla utrzymania produkcji urządzeń dających się eksportować. Kupować przy tym można to co trzeba, a produkując — szukać zastosowań dla tego, co się produkuje".*

●

Z całej dyskusji wyłania się obraz informatyki defensywnej, takiej, która podporządkowuje się i nie stara wywierać presji. To jest wprawdzie dzisiaj droga jedyna, niemniej najbliższe lata mogą zadecydować o miejscu Polski wśród krajów już choćby tylko średnio rozwiniętych. W. Turski: *„To jest fundamentalna rzecz. Czy możemy poczekać? Oczywiście, że możemy, ale to będzie nie do odrobienia już nigdy. Bo jeśli się tym teraz nie zajmiemy, będzie musiał nastąpić nie tylko import sprzętu i rozwiązań technologicznych, ale i import kultury, import cywilizacji".*

J. Kulikowski: *„Myśl prof. Turskiego powinna być przekazana do świadomości społeczeństwa, ale żeby nie była znowu odebrana jako szaleńcze wystąpienie jakiejś grupy nacisku, bo tak nas, niestety, ludzie odbierają. Kiedy mówimy, że jesteśmy 20 lat wstecz, że nie wiadomo czym się to dla nas skończy, może jakąś katastrofą społeczną czy narodową, ludzie spoza kręgu zawodowego informatyków posądzają nas, że dbamy o własny interes, że zabiegamy o więcej środków. Musimy o sprawach informatyki mówić społeczeństwu tak, aby chciano nas zrozumieć".*

Opracował Z. GŁUZA

P.1877/80 Cena 30 zł

KATOWICE, MAJ 1980 R.

AKT NADANIA

UCHWAŁĄ

KONFERENCJI SAMORZĄDU ROBOTNICZEGO
CENTRUM INFORMATYKI I BADAŃ EKONOMICZNYCH
HUTNICTWA

z dnia 23 kwietnia 1980 r.

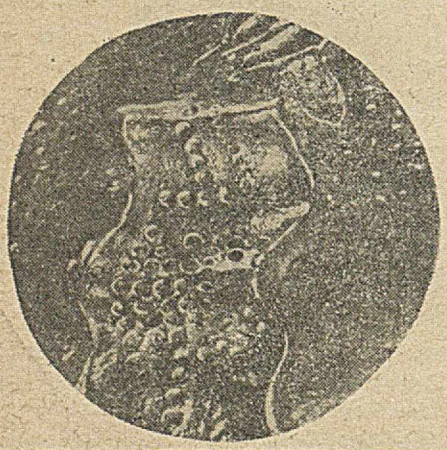
NADANO

REDAKCJI INFORMATYKI

PAMIĄTKOWY

MEDAL XXV-LECIA

CENTRUM INFORMATYKI I BADAŃ EKONOMICZNYCH
HUTNICTWA
W KATOWICACH



PRZEWODNICZĄCY KSR

(mgr inż. S. MOSZYK)

DYREKTOR

(Dr ZB. WINIARSKI)